Tutorial

LNAI 2086

Michael Luck
Vladimír Mařík
Olga Štěpánková
Robert Trappl (Eds

Multi-Ag
and Appl

9th ECCAI Advanced Course
3rd European Agent Systen
Prague, Czech Republic, Jul

Lecture Notes in Artificial Intelligence 2086

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Michael Luck   Vladimír Mařík
Olga Štěpánková   Robert Trappl (Eds.)

Multi-Agent Systems and Applications

9th ECCAI Advanced Course, ACAI 2001
and Agent Link's 3rd European Agent Systems
Summer School, EASSS 2001
Prague, Czech Republic, July 2-13, 2001
Selected Tutorial Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saabrücken, Germany

Volume Editors

Michael Luck
University of Southampton, Department of Electronics and Computer Science
Highfield, Southampton SO17 1BJ, UK
E-mail: mml@ecs.soton.ac.uk

Vladimír Mařík
Olga Štěpánková
Czech Technical University, Facultiy of Electr. Engineering, Dept. of Cybernetics
Technicka 2, 166 27 Prague 6, Czech Republic
E-mail: {marik/step}@labe.felk.cvut.cz

Robert Trappl
University of Vienna, Dept. of Med. Cybernetics and Artificial Intelligence
Freyung 6, 1010 Vienna, Austria
and Austrian Research Institute for Artificial Intelligence
Schottengasse 3, 1010 Vienna, Austria
E-mail: robert@ai.univie.ac.at

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Multi-agent systems and applications : 9th ECCAI advanced course ; selected tutorial papers / ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, July 2 - 13, 2001. Michael Luck ... (ed.). - Berlin ; Heidelberg ; New York ;Barcelona ; Hong Kong ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2086 : Lecture notes in artificial intelligence)
ISBN 3-540-42312-5

CR Subject Classification (1998): I.2.11, I.2, C.2.4, K.4, H.3, H.4

ISBN 3-540-42312-5 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10839338 06/3142 5 4 3 2 1 0

Preface

The Advanced Course on Artificial Intelligence ACAI 2001 with the subtitle “Multi-Agent Systems and Their Applications”, held in Prague, Czech Republic, was a joint event of ECCAI (the European Coordinating Committee for Artificial Intelligence) and AgentLink, the European Network of Excellence for Agent-Based Computing. Whereas ECCAI organizes two-week ACAI courses on different topics every second year, AgentLink’s European Agent Systems Summer School (EASSS) has been an annual event since 1999. This year, both of these important events were merged together, giving weight to the fact that multi-agent systems currently represent one of the hottest topics in AI research. The name, ACAI 2001 Summer School, is intended to emphasize that this event continues the tradition of regular ECCAI activities (ACAI), as well as the EASSS summer schools of AgentLink.

The Prague ACAI Summer School was proposed and initiated by both the Gerstner Laboratory, Czech Technical University, Prague (GL-CTU) and the Czech Society for Cybernetics and Informatics (CSKI), with the support of the Austrian Research Institute for Artificial Intelligence in Vienna (OFAI). Part of our motivation was catalyzed by experience gained in 1992 during the International Summer School “Advanced Topics in Artificial Intelligence” (see Springer’s LNAI vol. 617) which was organized by the same Czech and Austrian bodies.

One of the most important stimulating factors behind the organization of ACAI 2001 was the support provided by the European Commission to the Gerstner Laboratory within the frame of the MIRACLE Center of Excellence project (IST No. ICA1-CT-2000-70002). Additional support was later provided by both the Commission’s AgentLink II project (IST-1999-29003) and ECCAI, and the combined financial and conceptual participation of these important international bodies enabled the invitation of a large number of truly world-class lecturers in this field, who have added a unique flavor to the event.

In addition to the combined summer school, there were four co-located affiliated workshops/meetings: the AEMAS 2001 workshop (“Adaptability and Embodiment Using Multi-Agent Systems”), the ESAW 2001 workshop (“Engineering Societies in the Agents’ World”), the AgentLink II SIG meetings and the meeting of the FIPA Working Group on “Product Modeling and Manufacturing”. Thus, Prague became the “agent-world” capital in the first half of July 2001.

The main goal of the summer school was to present the current state of the art in the theoretical foundations of multi-agent systems as well as to demonstrate the applicability of these systems in many practical tasks. The choice of the topics and lecturers was driven by the desire to cover the field of multi-agent systems with the maximum breadth, while maintaining the utmost quality. As a result, the presentations highlight many different but complementary aspects and viewpoints of this recently established and very active scientific field.

The organizers also wanted to give the opportunity to Ph.D. students to briefly present the results of their ongoing work, to bring them to the attention of the distinguished experts in the field, and to provide a forum for valuable discussion and feedback with them. Thus, the summer school provides space for short presentations given by Ph.D. students within the frame of three students’ workshops. Forty student presentations were selected in a standard refereeing process, from which the best will be included – after some extensions and modifications – into the second, post-summer-school LNAI volume, together with selected papers from the accompanying AEMAS 2001 workshop and the remaining papers delivered by the invited speakers.

We would like to thank all the invited speakers for their willingness to contribute to the summer school and for their pro-active approach, as well as for delivering the promised manuscripts of their presentations in time. We understand that our intention to publish the invited lectures for a summer school in the form of a book is not usual , and we recognize that it may have required additional and unanticipated effort from the presenters, for which we are extremely appreciative. However, we wanted to share – in the form of this separate edited volume − the expertise of the invited lecturers with the wider AI and computing communities, and to provide essential readings to students, academics, and industrial researchers unable to attend the summer school.

Finally, we would like to thank the numerous collaborators who helped substantially to shape the basic ideas as well as to accomplish all the organizational and preparatory activities, namely Hana Krautwurmová, Jiří Lažanský, and Zuzana Hochmeisterová. Our thanks also go to Kamil Matoušek, who carried out the main portion of the computer work related to the preparation of both the camera-ready and electronic versions of this volume, and to Jiří Palouš who managed the ACAI 2001 website.

April 2001

Michael Luck
Vladimír Mařík
Olga Štěpánková
Robert Trappl

ACAI 2001

Ninth ECCAI Advanced Course

&

AgentLink's Third European Agent Systems Summer School

(EASSS 2001)

Multi-Agent Systems and Their Applications

Prague, Czech Republic, July 2-13, 2001

Program Co-chairs:

Michael LUCK	University of Southampton, UK
Vladimír MAŘÍK	Czech Technical University, Czech Republic
Olga ŠTĚPÁNKOVÁ	Czech Technical University, Czech Republic
Robert TRAPPL	Austrian Research Institute for AI, Austria

Invited Speakers:

Hamideh AFSARMANESH	University of Amsterdam, The Netherlands
Elisabeth ANDRE	DFKI GmbH, Germany
Luis M. CAMARINHA-MATOS	New University of Lisbon, Portugal
Misbah S. DEEN	University of Keele, UK
Yves DEMAZEAU	Leibnitz-Institute IMAG, France
Jim DORAN	University of Essex, UK
Edmund H. DURFEE	University of Michigan, USA
Klaus FISCHER	DFKI GmbH, Germany
Les GASSER	University of Illinois, USA
Jozef KELEMEN	Silesian University, Czech Republic
Matthias KLUSCH	DFKI GmbH, Germany
Sarit KRAUS	University of Maryland, USA
Yannis LABROU	University of Maryland, USA
Jörg MÜLLER	Siemens, Germany
Bernhard NEBEL	University of Freiburg, Germany
Eugénio OLIVEIRA	University of Porto, Portugal
Paolo PETTA	Austrian Research Institute for AI, Austria
Stefan POSLAD	University of London, UK
Katia SYCARA	Carnegie Mellon University, USA
Milind TAMBE	University of Southern California, USA
Paul VALCKENAERS	Catholic University of Leuven, Belgium
Wolfgang WAHLSTER	DFKI GmbH, Germany
Hendrik VAN BRUSSEL	Catholic University of Leuven, Belgium
Wiebe VAN DER HOEK	Utrecht University, The Netherlands
Michael WOOLDRIDGE	University of Liverpool, UK

Contributors:

Olaf BOCHMANN	Catholic University of Leuven, Belgium
Patricia CHARLTON	University of London, UK
Kurt DRIESSENS	Catholic University of Leuven, Belgium
Petra FUNK	DFKI GmbH, Germany
Dimitar KAZAKOV	University of York, UK
Martin KOLLINGBAUM	Catholic University of Leuven, Belgium
Daniel KUDENKO	University of York, UK
Vladimír KVASNIČKA	Slovak Technical University, Slovakia
Michal PĚCHOUČEK	Czech Technical University, Czech Republic
Jiří POSPÍCHAL	Slovak Technical University, Slovakia
David V. PYNADATH	University of Southern California, USA
Christian RUSS	DFKI GmbH, Germany

Organizing Committee:

Hana KRAUTWURMOVÁ	Czech Technical University, Czech Republic
Jiří LAŽANSKÝ	Czech Technical University, Czech Republic
Zuzana HOCHMEISTEROVÁ	Czech Technical University, Czech Republic
Eva KUBRYCHTOVÁ	EKU Agency, Czech Republic
Jiří PALOUŠ	Czech Technical University, Czech Republic

Table of Contents

Foundations of Multi-agent Systems

Perspectives on Organizations in Multi-agent Systems 1
L. Gasser

Multi-agent Infrastructure, Agent Discovery, Middle Agents for
Web Services and Interoperation 17
K. Sycara

Logical Foundations of Agent-Based Computing 50
W. van der Hoek

Standardizing Agent Communication 74
Y. Labrou

Standardizing Agent Interoperability: The FIPA Approach 98
S. Poslad and P. Charlton

Distributed Problem Solving and Planning 118
E.H. Durfee

Automated Negotiation and Decision Making in Multiagent Environments 150
S. Kraus

Agents' Advanced Features for Negotiation and Coordination 173
E. Oliveira

Social Behaviour, Meta-reasoning, and Learning

Towards Heterogeneous Agent Teams 187
M. Tambe and D.V. Pynadath

Social Knowledge in Multi-agent Systems 211
V. Mařík, M. Pěchouček, and O. Štěpánková

Machine Learning and Inductive Logic Programming for Multi-agent
Systems 246
D. Kazakov and D. Kudenko

Relational Reinforcement Learning 271
K. Driessens

From Statistics to Emergence: Exercises in Systems Modularity 281
J. Kelemen

Emotions and Agents 301
P. Petta and R. Trappl

Applications

Multi-agent Coordination and Control Using Stigmery Applied
to Manufacturing Control 317
P. Valckenaers, H. Van Brussel, M. Kollingbaum, and O. Bochmann

Virtual Enterprise Modeling and Support Infrastructures:
Applying Multi-agent System Approaches 335
L.M. Camarinha-Matos and H. Afsarmanesh

Specialised Agent Applications 365
K. Fischer, P. Funk, and C. Ruß

Agent-Based Modelling of Ecosystems for Sustainable Resource Management . 383
J. Doran

Cooperating Physical Robots: A Lesson in Playing Robotic Soccer 404
B. Nebel

A Multi-agent Study of Interethnic Cooperation 415
V. Kvasnička and J. Pospíchal

Author Index 437

Perspectives on Organizations in Multi-agent Systems

Les Gasser

Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
501 East Daniel St., Champaign, IL 61820
gasser@uiuc.edu
http://www.uiuc.edu/~gasser

Abstract. The aim of this paper is to illustrate and sensitize reader to the variety of perspectives and the fundamental nature of organizations as stable/stabilizing systems and as multi-perspective action systems. Researchers have been explicitly thinking about MAS/DAI organizations and attempting to link formal (human) organization theory with MAS/DAI models for at least twenty years. Despite this, the idea of organizations has been a peripheral theme in MAS/DAI research---primarily a specific coordination technique---not really one of the central intellectual issues of the field. The theory of 'natural' organizations has a somewhat longer, more diverse, and more thorough intellectual history than that of organizations in MAS. Beyond recent work in human social and organization theory, some newer research on abstract organizations has been attempting to unify concepts in biology, chemistry, physics, mathematics, and computing theory (e.g., the lambda calculus), with those of natural social organizations and multi-agent systems. The landscape for thinking about organizations in MAS is growing quite interesting, and this paper surveys this landscape. It presents and contrasts some conceptions of organization that have emerged and proven useful, and attempts to show how these have been implemented, experimented with, and applied. It also projects some future directions for research on MAS organizations, and gives some thoughts on where the most exciting issues lie.

1 Introduction

The aim of this paper is to illustrate and sensitize reader to the variety of perspectives and the fundamental nature of organizations as stable/stabilizing systems and as multi-perspective action systems. It's not news that information and computation are rapidly becoming ubiquitous and embedded parts of modern life in all spheres. To name just a few:

- The physical environment: smart buildings, roads, buildings, nanomachines, and 'smart matter'
- Critical infrastructures: energy/power distribution, telecommunications, transportation/logistics, intelligent HVAC, digital libraries, etc.
- Healthcare: biocomputing, patient records, non-invasive instruments

- Daily life: wearable computing, home networks/smart houses, personal information management.
- Science and engineering: embedded instrumentation, large-scale automatic instrumentation and data gathering, distributed experimentation, computational grids.

Establishing the principles and tools for understanding and organizing this growing fabric of information and computing is one of the next critical frontiers for the information and computer sciences.

With the growth of this fabric, especially those parts driven or inspired by the Internet, interest in high-level, modular, and data/knowledge-rich computation frameworks including client-server frameworks, peer-to-peer interactions, and 'agents' is exploding. Many people share a vision of ubiquitous embedded computing, information webs, knowledge networks, digital libraries, or electronic commerce environments that are very active, populated with roving, dynamic information management agents and teams. They imagine the capability of easy, seamless integration of heterogeneous modular units: teams of agents forming and reorganizing at will, to respond to new information management and retrieval tasks, new economies of information, new ways of consolidating information. etc. They imagine agent collections easily integrating new 'team members' and re-coordinating themselves for robust, efficient performance as new services, skills or operating conditions appear.

And this vision doesn't stop with information-oriented environments and applications---action in the world is also part of the newest scenarios. The vision of "information appliances" promulgated by Hewlett-Packard, Sun, Cisco, Microsoft, and others through technologies such as "Javaspaces," "Jini" and "Universal Plug and Play" includes physical appliances such as toasters and HVAC, entertainment appliances, and so on as active, self-organizing components of a huge, interactive network-based fabric.

2 Critical Barriers and Knowledge Gaps

In large measure, the systematic scientific principles and robust coordination technologies that will help this vision to unfold effectively and with comprehensible, predictable operational character don't exist. From a scientific standpoint, there are a number of barriers. From the standpoint of this paper---organizations in MAS---the critical barriers include the following:

- We have relatively limited principled understanding of how to organize sophisticated, interdependent, heterogeneous, semi-autonomous computational objects--and the infrastructures to support them---into aggregates with stable, predictable, and reliable behavior at a very large scale.
- More than simply achieving stable reliable behavior in collectives, the point of formal MAS organizations is to achieve economies of scale and scope by exploiting features such as division of labor, integrated coordination, and distrib-

uted commitment. We have limited principled knowledge of how to create and exploite the theories and techniques that will create such added efficiencies, yet it is absolutely essential to do so as the rate of resource consumption (e.g., the resources of information search, interpretation, assimilation, application, communication, etc.) increases faster than our ability to provide it.

- We lack the practice and experience in building and operating such organizational systems in situ.

From a scientific standpoint, the foundations of this vision have a long history. For over thirty years, a multidisciplinary community of researchers and developers has been exploring new paradigms of information, knowledge, problem-solving, and activity that are based on group, team, organizational, and social-level approaches. This research uses computational models, metaphors, representations, and research tools to build and test theories and mechanisms for dynamic, coordinated, interactive computing. It has incorporated (and added to) knowledge from fields including information/computer science, economics/game theory, organization theory/social science, and even ethology to name a few. Over time, the subjects of this research field have been variously called "Agents", "Multi-Agent Systems" (MAS), "Coordinated Computing," "Coordination Theory," "Distributed AI," and so on.

The thrust of this paper is to begin to illustrate some of the issues that should be placed at the forefront of thinking about organizations as a critical issue for the future of MAS. Researchers have been explicitly thinking about MAS/DAI organizations and attempting to link formal (human) organization theory with MAS/DAI models for at least twenty years. Despite this, the idea of organizations has been a peripheral theme in MAS/DAI research---primarily a specific coordination technique---not really one of the central intellectual issues of the field. The theory of 'natural' organizations has a somewhat longer, more diverse, and more thorough intellectual history than that of organizations in MAS. Beyond recent work in human social and organization theory, some newer research on abstract organizations has been attempting to unify concepts in biology, chemistry, physics, mathematics, and computing theory (e.g. the lambda calculus), with those of natural social organizations and multi-agent systems. The landscape for thinking about organizations in MAS is growing quite interesting, and this paper surveys this landscape. It presents and contrasts some conceptions of organization that have emerged and proven useful, and attempts to show how these have been implemented, experimented with, and applied. It also projects some future directions for research on MAS organizations, and gives some thoughts on where the most exciting issues lie.

3 Perspectives on Organizations in MAS

There are three orientations commonly taken with respect to organization in MAS research: Theoretical, phenomenological, and technological.

Theoretical Orientation: A *theoretical orientation* attempts to define and delimit the general concept of organizations, to establish taxonomies and varieties of possible organizations, and to make general statements about organization, organizing processes, and organizations, in the abstract. From this orientation, organizations may or may not exist a priori. The properties of organizations as abstract types is the main point of interest. The information processing aspects of organization and organizing are, then, very compatible with computational approaches to theorizing about organizations, because both can be treated productively using common abstractions.

Phenomenological orientation: A *phenomenological orientation* attempts to describe, model, and explain existing, observable organizations as phenomena to be accounted for. From this viewpoint, organizations exist, and researchers need to explain how and why they come into being and persist. Phenomenological theories attempt to account for the observed properties of naturally-occurring organizations. Since phenomenologists build models, computational tools and modeling techniques are useful, even to the extent that they predict or explain fundamental limits and thus help to account for the actual behavior or structure of observed organizations. But for phenomenologically inclined researchers, the ending point is what can be observed in the natural world, not what are the abstract theoretical possibilities for organizing.

Technological Orientation: A number of organization scientists and economists have argued that natural organizations (as well as other social forms) come into being as large-scale problem solving technologies to deal with circumstances that otherwise couldn't be handled because of the limitations of individual actors [44], [29], [6]. (Note that this class of explanations typically assumes a prior existence and character of individual agents from which organizations are comprised, and whose limitations are addressed by organizing. It is essentially a reductionist viewpoint in that it doesn't account for organizations themselves as fundamental entities, nor for agents that emerge from collectives, nor for agents and collectives that mutually construct each other in a "circular" fashion. See, e.g., [23], [24], also see below). Limitations to be overcome by organizing may be of several types:

Cognitive Limitations: bounds to the rationality or computational tractability of information processing in individual agents create pressures for collective action to achieve ends with more extensive cognitive or computational requirements.

Physical Limitations: Bounds on physiology, resources, and access across space and time create pressures for comunication and binding among multiple agents to extend action, perception, and access to resources across broader regions of a space that is accessible to individuals.

Temporal Limitations: Bounds on agent lifetimes create pressures for collective action across time when the temporal span of action extends beyond the lifetimes of agents.

Institutional Limitations: Individual actors are limited in legal, political, economic, and other institutional senses. Collective action can modify the insitutional status of individuals and can create new institutional-level actors with status specialized for institutional-level tasks [6], [37], [42].

Large construction projects, for example, require moving amounts of material over great distances---needs that are far beyond the capabilities of individuals alone. Economies of scale and scope gained by organizing activity through space and time are what make large projects possible. Thus, technological orientations see organization and organizing processes fundamentally as technologies to achieve ends such as efficiency, scale-of-effort, complexity management, or human systems integration. From this viewpoint, organizations are a kind of social technology that can be put to use in selected contexts to accomplish some designable but supra-individual end.

This set of three orientations---the theoretical, the phenomenological, and the technological---begins to give us a handle on how to frame existing MAS research on organizations by considering the goals and assumptions of the researchers and the kinds of issues that can be understood. They are not necessarily mutually exclusive. For instance, organizational phenomenologists create models and explain phenomena, and these models and explanations may draw n abstract theory or on technological explanations.

4 Analysis of Critical Organizations Concepts

We will not attempt to give a precise or exhaustive definition of organizations here, but instead will aim to capture a number of characteristics that are on the one hand, typical of organizations from an MAS perspective, and on the other hand, useful in recognizing, talking about, theorizing about, and building MAS organizations. First, organizations are structured, patterned systems of activity, knowledge, culture, memory, history, and capabilities that are distinct from any single agent. We say, then, that organizations and many organizational phenomena and characteristics are "supra-individual" phenomena: they exist at a level independent of specific individual behaviors or attributes, such as an aggregate level or as a type. We also insist on the fact that organizational explanations should be based on organizational (that is aggregate or other supra-individual) information or models. Organizations have relationships with other social[1] entities such as institutions and social groups. Finally, organizations have some "extent" across some abstract space. That is, they "take up" or "occupy" some region of a defining space such as geographical space (i.e., a physical dimension of organization), time, semantics, symbols, deduction, and so on. Several concepts are basic to considering organization. Not all of these concepts apply as clearly or as essentially to all organizations. Nonetheless, each of them serves to sensitize us to critical issues in thinking about organizations, especially from general, dynamic, and computational perspectives.

[1] In this paper, the term "social" means specifically a) occurring at multiple space-time locations simultaneously, b) involving multiple perceptual/analytical perspectives, and c) incorporating internal and external interdependencies and linkages. That is, "social" things happen concurrently in different places, they involve many different viewpoints or perspectives simultaneously, and their parts are linked and hence constrained both inwardly and outwardly. This is an abstract, structural sense of "social", not one specific to people, and it explicitly doesn't refer to affective dimensions typical of many human social activities. These are nonetheless important, especially in the context of organizations that integrate people and machines (e.g. [40]). The requirement of "simultaneous multiple perspectives" is, for me, the most critical defining characteristic.

Division of activity types: activity in organizations is not uniformly or randomly distributed across an organization's defining space. Instead instead types of activity are differentially distributed, leading to the concept sometimes called "division of labor" for human organizations. Phenomenologically, we might say that activities can be segmented by type, and different activity types may be observed to occur with differential frequency in different regions of an organization's defining spaces. Technologically speaking, we can design organizations to exploit eficiencies derived from appliing differential skills of participants and different stable mappings of skill sets to organizational goals or problems.

Integration: Organizations have interdependencies between different regions of activity in their defining spaces. That is, an organization is in some sense a set of constraints on the set of relationships between points in the organization's defining spaces. In addition, organizations are "whole" systems of information and activity, not just unstructured collections of parts and localized actions. Dimensions of organization are integrated, in the sense that they are aligned and interacting, not random and separate.

Compositionality: Organizations are structures that are composed of other structures through a variety of stabilizing and aggregative mechanisms (cf. [14]). However, the assumption that organizations necessarily comprise composable atomic units such as "agents" or "individuals" isn't a fundamental aspect of organization. I take this position for two reasons: First, the notion of organization is fundamentally a supra-individual, collective notion. Properties of organizations that are properly termed "organizational properties" only exist at the aggregate level. Second, the notion of circular causality means that individuals are the product of organizing processes, both in a "bottom up and top down" sense, and in a lateral, collective sense [23]

Stability/Flexibility: Organizations exhibit patterns of activity and in fact one way to describe organizations both theoretically and phenomenologically is as observable patterns of activity[2]. These patterns have stable features (the stable features is what makes them patterns) and they have flexible features (no two collections of organizational activity are ever the same, and in fact flexibility of action is important in dynamic environments). In effect, then, organizations can be seen as architectures: stable, constraining structures within which a certain degree of flexible activity is possible. Generalizing to multiple dimensions of organizational description, another way of saying this is that organizations are "specific collections of settled (stable) and unsettled (flexible) questions" (see below and [21]).

Coordination: If organizations are to effect economies of scale and scope, they must have efficiencies beyond the simple aggregate sum of the efficiencies of individual actors; they must be, in effect, resource amplifiers, increasing order within their operational scopes. [2], [49], [28]. These efficiencies are made difficult due to the combined effects of interdependency and uncertainty. Collections of interdependent activities with multiple possible outcomes create regions of activity space with differential value. Efficiencies require optimization to activity patterns that fall into

[2] From the technological perspective, the core problem is how to link desired aims to specific mechanisms that create appropriate, efficient patterns of activity---how to get the "right" patterns of activity.

the highest-value regions, and are sustained there. Unfortunately, with uncertain local information about possible distributed outcomes, and with limited ability to affect distributed activities at-a-distance, convergence to high-value activity regions is difficult. In physical systems, interactional uncertainty is manifested through the statistical mechanics of dynamic multi-body systems, yielding aggregate-level relationships and "laws." In systems where aggregate behavior measures are too gross to achieve the efficiencies required, subtler and more localized mechanisms are needed. This is the role of coordination techniques.

Supra-Individuality and Roles: Organizations are certainly specific, concrete activity systems that enact specific concrete actions in specific, situated locations and contexts. But, paradoxically, they are also activity systems structured through complex sets of activity types. The internal linkages that "glue" and stabilize organizational activity are founded on actors' ability to generalize from specific instances of interactive behavior to prototypical types of behavior, to develop interlocking commitments to and expectations of those behavioral types, and finally to actualize or realize new instances of joint activity that correspond with those types. The repeated re-enactment of activity of known types reinforces the structure of the organization in ongoing ways, and the organization continually remakes itself in action. (See Figure 1.) Weick has used the term "double interact" to describe a patterned, interlocking dyadic (2-agent) activity unit, from which he suggests all organizations are built [51]. Similarly, Giddens notes the continual process of reconstitution that builds social collectives [24]. Similar cycles of autocatalytic reproduction are theorized as foundations of biological self-organization [12], [14].

A key aspect of the generalization, typing, and realization cycles that constitute organizations is that they are collectively consensual. The generalizations to activity types are fundamentally organizational phenomena, not agent-level phenomena, in the sense that no single agent can assert a specific generalization or 'meaning' of activity, and hence no single agent can assert or sustain a commitment or expectation that involves such a generalized activity type. The understanding and expectations built into these activity types are spread through the organization. What this means is that organizations are supra-individual---they exist above and beyond single individual agents that may constitute them---and agents are replaceable into and out of types.

Another key aspect of activity types is that they capture the generalized properties of linkages among activities. All organizational activities are structured and constrained. These structures and constraints are also generalizable into types. To describe an activity type, it is also necessary to describe the "standard" set of linkages between the activity type and other activity types (and indeed linkages to other organizational aspects besides activity, such as information, resources, context, etc.) This means that activity types are inherently linked or distributed entities as well.

Another word commonly applied to a generalized activity type is "role" (e.g., [33]). Role typically describes an organizationally-sanctioned (i.e., consensually defined) structured bundle of activity types. Since an activity type naturally incorporates linkages and constraints that connect instances of it to other organizational aspects, it is natural to see a role as the abstraction of a position in a stable (i.e. generalized, repeatedly reinforced) set of relationships. This an the approach taken by Social Network theorists [50]. Roles are thus clearly supra-individual constructs because they consist of activity types, not to concrete, specific individuals, which can be replaced.

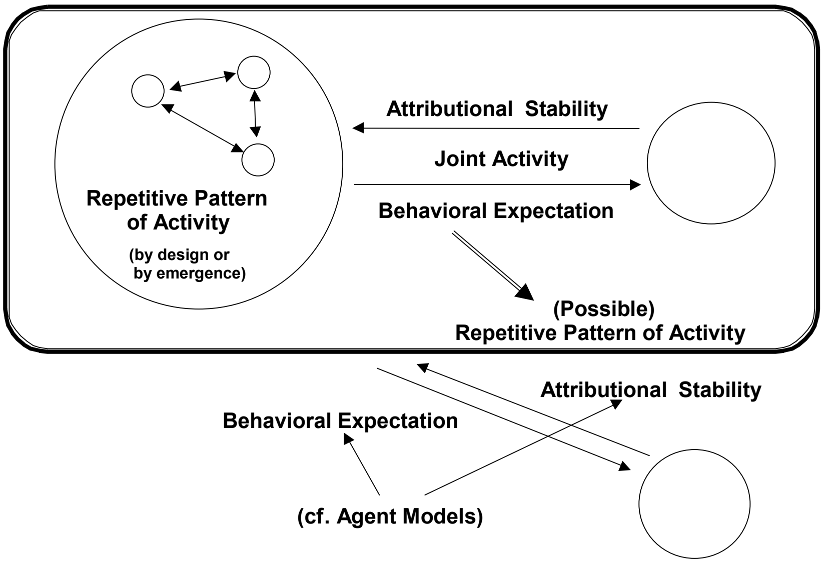


Fig. 1: Organizations as Activity Patterns

Recursivity: Organizations are often composed of sub-organizations, and/or of multiple overlapping structures. In particular, the cascaded repetitive patterns of activity depicted in Figure 1 appear at multiple levels of analysis. This means that the phenomenology of organizations has a recursive character, and we would like theories and technologies of organization also to account for and make possible recursively structured collectives [45], [14]

Multi-Level Representation and Causality: Organizations comprise multiple overlapping multi-level structures. The activities which constitute organizations form patterns in multiple scales of time, space, and other dimensions. Hence the patterned structures of organization interact with each other, and they interact with the environment of the organization, at multiple levels. Because of this, causality in organizations is a multi-level phenomenon. For example, in Durfee, Lesser, and Corkill's classic model [11], [8], organization is seen as a long-term temporal commitment to sets of activities. These activities are also organized into short term reactive cycles and medium-term planning cycles, each of which is contextualized by the longer-term cycles that embed it. Thus concrete behavior emerges from three interlocking levels of repetition, and can be affected by changes in any of these levels. New technologies for experimenting with agent-based computational organizations also explicitly capture both nested temporal cycles and interlocking, multilevel activity patterns. The creators of the RePast modeling testbed, for example, describe it like this:

"The name Repast is an acronym for REcursive Porous Agent Simulation Toolkit. Our goal with Repast is to move beyond the representation of agents as discrete, self-contained entities in favor of a view of social actors as permeable, interleaved and mutually defining, with cascading and recombinant motives. We intend to support the modeling of belief systems, agents, organizations and institutions as recursive social constructions."

The MACE3J/TaskModel platform [19] also incorporates multilevel, recursively modeled structures of mutually-defining activities. On the theoretical side, see also [4], [14], [52].

Potentials and differentials: In physical systems, particular balances of attractive and repulsive forces yield particular organizational structures and dynamics. Ferber, among others, has exploited this simple attraction/repulsion ordering technique in experiments [13] with interesting results, and it is a foundation of many abstract models of self-organization systems [34]. Organizational actors shape their activities with varying degrees of information and varying degrees of influence on other actors. In addition, the cycles of generalization, interaction, and realization noted above create differentials of expectation, commitment, and flexibility as generalizations have wider or narrower scope, or are more or less strongly reinforced through re-enactment. Such differentials affect how activities can be carried out and how interdependencies can be managed. They also lead to singularities and non-uniformities in structure that can take hold and serve as organizing anchor points [48]. Power, for example, is generally described as the ability to influence the behavior of an organizational unit or organizational actor. In general, organizational patterns are mediated by many such force differentials.

Rules and Grammars: It is a commonplace in the theoretical and phenomenological literature that organizations are rule-governed structures [30], [32]. More recently, some theorists have begun to consider organizations as individual points in spaces of potential configurations of activity and process. They have begun describing these configuration spaces using grammars, sometimes with and sometimes without the attachment to some kind of "deep structure" that is common in linguistic grammatical theories [35], [36], [39]. Rules have three complementary interpretations in the theoretical literature on organizations:

- Rules as structures for action--that is, as procedural specifications
- Rules as constraints on action--that is as architectures of what is possible and what is proscribed
- Rules as compiled experience [32]

Grammars of organization are also interpreted under these three views. Ideally, a rule-based or grammar-based concept (and even implementation) of organization would make computational treatments of organization all the easier, as the theoretical and technological base for them is strong in computing. Unfortunately the dynamic, emergent, evolutionary, and situational aspects of both organizational grammars and organizational rules are also interesting and not well investigated. Manning, for example [30] presents a compelling analysis of high fluidity in the practical application of rules in organizations. In his view this fluidity comes about due to situational aspects of rule interpretation: all rules need interpretation in their application and the situated nature of their interpretations undermines their strategic, stabilizing, and patterned aspects (not to mention their potential status as "organizational procedures"). This disruptive in-situ character is a fundamental aspect of all distributed information semantics, not just of rules [17]. However, it means that the account of rules and grammars as part of the fundamental character of organization needs expansion.

Uncertainty: Distribution of information necessarily introduces uncertainty into an information-mediated action space. The uncertainty is due to limitations of communication as well as to the concurrent nature of organizational activity. In addition, numerous systems exhibit inherent stochastic properties at the collective and individual levels. The effect of uncertainty in organizational systems is to limit the ability for activity systems to guarantee convergence to regions of activity space that exhibit high value. Thus, many technological strategies for organizing take the approach of expressly managing uncertainty as a coordinating function. This creates requirements for technologies (and theories) for uncertainty capture, representation, and reduction, in organizational systems. Chemical catalytic systems, for example, exploit the ability of catalysts to physically orient several molecules toward each others' binding sites. This reduces the uncertainty that reactions will take place (i.e. it increases reaction probabilities) and forces the aggregate reaction behavior toward an equilibrium more saturated with reaction products than without catalysis (cf. [12], [14]). Analogously, MAS organizational models that expressly deal with uncertainty (e.g., those of V. Lesser, N. Carver, and T. Wagner) attempt to focus global effort on most promising directions by reducing probabilities of poor local choices. This is accomplished via communication of intentions and meta-level information, and by local dynamic adaptation as new sources of uncertainty (or new information that reduces uncertainty) is discovered in real time.

With these analyses of organizations concepts in mind, we turn to the most productive current arena for studying and developing them further: Computational Organization Research.

5 Computational Organization Research (COR) as a Venue for Studies of MAS Organizations

Computational methods may provide several advantages for studying and enhancing organizational phenomena. Organization theory, analysis and design problems can be hard because of the scale and complexity of both the objects of study (organizations and organizational processes) and of the theories themselves. The epistemological, structural and configuration problems of organizations are at least as impactful as analogous problems that appear in other disciplines that regularly employ computational methods in similar ways. For the study of organizational issues, the foundation of theory, modeling technology and infrastructure is ready, and the impact of improved knowledge and effectiveness and flexibility of organizations could be quite significant.

Research in this area draws on work in distributed artificial intelligence (DAI), multi-agent systems, adaptive agents, organizational theory, communication theory, social networks, and information diffusion. One of the foundational works in this area is The Behavioral Theory of the Firm [10] in which a simple information processing model of an organization is used to address issues of design and performance. While the strongest roots are in the information processing literature [43], [31], [47], [16], [10] and social information processing traditions, current models also have roots in the areas of resource dependency, institutionalism [37], [42], and symbolic interactionism [17], [19]. Formalisms and specific measures of organizational design are drawn from the work in the areas of coordination [29], social networks [50] and distributed control [9], [11].

Computational Organization Research (COR) can be examined along three axes:

Computational: The focus of this activity is explicitly computational approaches to organizational phenomena, including models and representations of organizational features and concepts such as those treated above. Of particular relevance to MAS, COR focuses on computational organizations, namely those made up entirely of computational entities such as agents and mult-agent aggregates.

Organization: The locus of this activity is typically both mid-range organizational-level phenomena in which aggregate, statistical models and theories take a back seat to detailed, behavioral accounts, and macro-scale phenomena that rely heavily on aggregate and statistical properties and behaviors. This line is sometimes hard to draw, and the issue of macro-mezzo-micro links is often of explicit theoretical and practical interest.

Research: The locus of COR activity is primarily organizational research, that is, innovations in description, analysis, theory, and methods. It is clear that much good organization research, especially that which occurs under the phenomenological and technological frames described above, is driven by clear applied problems, and that the best practical organization analysis tools embody clear principles and theory. This line, too, is hard to draw, and COR may include issue in the practical application of organizational tools.

COR is also timely at this point due to a convergence of several trends in computing and in organizations. In the aggregate, the impact of these trends is shown in Figure 2 and described below. Main points of convergence include:

- New kinds of information-based, networked organizations such as virtual organizations provide new classes of study objects (literally, "computational organizations" and new simulation, experimental, and information gathering technologies.

- Representation and implementation technologies and theory have advanced to the point where it is possible to capture complex organizational fundamentals such as those referenced above.

- Advances in computer-aided design have created the tools to bring computer-aided organization design into reach. Computational modeling and evaluation of organization can support progress on computational organization design. Techniques of qualitative modeling, optimization, and search developed for other applications also can support computational organization design [22], [26], [35].

- High-powered desktop computing creates demand for generally-available tools for organization monitoring, analysis, and design. These machines along with very fast high-performance simulations capabilities on supercomputers [19], [20] add the capability to do complex analyses of very large systems with reasonable response times ***FN For example, using the MACE3J system we have been modeling multi-agent organizations that enact complex, dynamic task networks. We routinely run simulations of 5000 agents, 10,000 tasks, 500 problem instances, and ca. 10 million messages [19].

- Critical infrastructure also exists in the form of research collaborations among working groups, and transferable analytical and modeling software and data.

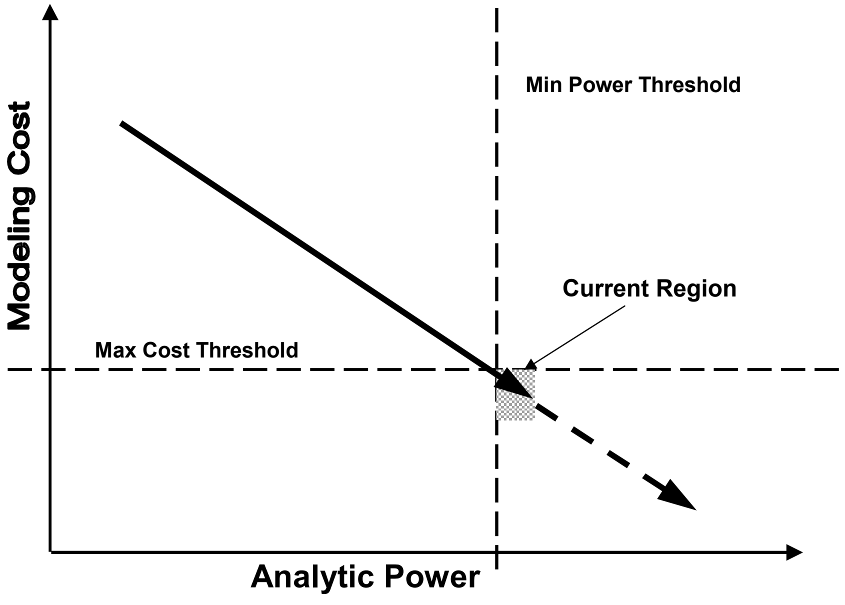


Fig 2.: Computational Organization Modeling Prospects

Figure 2 illustrates the current state of progress schematically. To be successful and useful, the ability to capture and model organizational phenomena such as those analyzed above should satisfy two thresholds. First, computational models must offer a degree of analytical power that places them in a different class of method than what was previously available. We would like to see COR models offer insights that are unachievable with other means. Second, the entry and use costs of COR approaches and models should be low enough to enable a critical mass of collaborating practitioners to develop communities of practice in research. This is because the scale of phenomena, models, data, and indeed the complexity of the conceptualizations of COR phenomena creates tremendous need for collaboration in COR research. Figure 2 illustrates the continuing trajectory of progress for COR models, from regions of high cost and low payoff, to the current state of marginally useful concepts, models, and infrastructure. Progress should occur rapidly as models and techniques proliferate from this point onward.

Another feature of current activity is the need to raise awareness of several fundamental assumptions of COR models, as follows. To be rational, COR researchers generally subscribe to the following assumptions [6]:

- ***Modelability:*** Organizational phenomena are modelable.
- ***Performance differential:*** It is possible to distinguish differences in organizational performance.
- ***Manipulability:*** Organization are entities that can be manipulated and transformed.
- ***Designability:*** Organizations are entities that can be designed. This is not to say that organizations do not evolve, nor that they cannot be found in nature, for assuredly both events occur. However, they can also be consciously designed and redesigned: organizational transformations can be purposeful and principled.
- ***Practicality:*** Organizational transformations (based on the design or manipulation of models) can be transferred into and implemented in actual practice.
- ***Pragmatism:*** The costs of modeling and researching organizations using computational methods are relatively lower than the costs of manipulating or researching similar aspects of actual organizations in vivo, and the benefits gained outweigh the costs.

Over time, these assumptions are becoming better defensible, and the degree of activity in COR circles is growing. COR provides a particularly effective venue for development of both theory and practice of organizations in MAS.

6 Conclusions

Thinking about organizations in MAS is an area of developing importance. Several perspectives have been presented, along with a comments on a number of basic themes and issues in multidisciplinary approaches to reasoning about organizations. In the short term, the most productive direction for research into organizations and MAS is likely to be that work that can be applied in the analysis, modeling, and simulation of organizations. Such work will have at least five benefits (cf. [20]):

- it will help advance the computational theory and conceptualization of organizations
- it will develop technical methods and infrastructure for implementing and experimenting with MAS organizations
- it will develop researchers' practice and experience with MAS organizations
- it will improve the pedagogical infrastructure and technology for MAS
- it will develop results on organizational models from human sciences, raising the impact of MAS research in the short term.

7 Acknowledgements

My recent thinking on the issues discussed in this paper has benefited greatly from conversations with Jean-Pierre Briot and Vic Lesser, and from the participants in my seminar on Self-Organizing Information Systems, Spring, 2001. This work has been supported in part under NSF Grant 9981797.

References

This section includes references to works directly cited in the text, as well as works that bear on the ideas in the text in a general way. The latter are included to support further investigation of associated subjects by the reader.

1. Philip Anderson. "Complexity Theory and Organization Science" Organization Science, Volume 10, Number 3, 1999.
2. W. Ross Ashby. "Principles of the Self-Organizing System", in: Principles of Self-Organization, von Foerster, H. & Zopf G.W. (eds.), (Pergamon, Oxford), p. 255-278, 1962.
3. Alan H. Bond and Les Gasser, editors. Readings in Distributed Artificial Intelligence. San Mateo, CA: Morgan Kaufmann, 1988.
4. Leo Buss. The Evolution of Individuality . Princeton, NJ: Princeton University Press, 1987.
5. Kathleen M. Carley and Michael J. Prietula, editors. Computational Organization Theory. Lawrence Erlbaum Associates. Hillsdale, NJ, 1994.
6. Kathleen Carley and Les Gasser, "Computational Organization Research," in Gerhard Weiss, ed., Distributed Artificial Intelligence", MIT Press, 1999.
7. Michael D. Cohen, James G. March and Johan P. Olsen. "A Garbage Can Model of Organizational Choice." Administrative Science Quarterly, 17(1): 1--25, 1972.
8. Daniel D. Corkill and Susan Lander. "Diversity in Agent Organizations," 1998. http://www.bbtech.com/papers/organizational-diversity.html

9. Daniel D. Corkill. A Framework for Organizational Self-Design in Distributed Problem-Solving Networks, Ph.D. Dissertation, Dept. of Computer Science, University of Massachusetts, Amherst, 1983.
10.Richard Cyert and James G. March. A Behavioral Theory of the Firm. 2nd Edition. Blackwell Publishers, Cambridge, MA, 1992[1963].
11.Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. "Coherent Cooperation Among Communicating Problem Solvers." IEEE Transactions on Computers, C-36: 1275-1291, 1987.
12.Manfred Eigen and Peter Schuster. The Hypercycle: A Principle of Natural Self-Organization. Springer, Berlin, 1979
13.Jacques Ferber. "Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence". Addison-Wesley, 1999
14.W. Fontana and Leo W. Buss, "The barrier of objects: From dynamical systems to bounded organizations," in: Boundaries and Barriers, J. Casti and A. Karlqvist (eds.), pp. 56-116, Addison-Wesley, 1996
15.Mark S. Fox. "An Organizational View of Distributed Systems," pp. 140-150 in Bond, Alan H. and Gasser, Les (eds.) (1988) Readings in Distributed Artificial Intelligence (San Mateo, CA: Morgan Kaufmann).
16.Jay Galbraith. Designing Complex Organizations. Addison-Wesley, Reading MA, 1973.
17.Les Gasser "Social Knowledge and Social Action: Heterogeneity in Practice," Invited Paper, in Proceedings of the 1993 International Conference on Artificial Intelligence (IJCAI-93), Chambery, France, pp. 751--757, 1993.
18.Les Gasser, "Computational Organization Research." International Conference on Multi-agent Systems, San Francisco, June 1995.
19.Les Gasser, "MACE: an Advanced, Integrative Infrastructure Supporting the Science of MAS." Working Paper LG-2001-04, GSLIS, University of Illinois at Urbana-Champaign, 2001.
20.L. Gasser, "MAS Infrastructure Definitions, Needs, Prospects,' in T. Wagner and O Rana, (eds.), Infrastructure for Scalable Multi-agent Systems, Lecture Notes in Computer Science, Springer, 2001.
21.Les Gasser, Nicholas Rouquette, Randall W. Hill, and Jon Lieb, "Representing and Using Organizational Knowledge in Distributed AI Systems." In Les Gasser and Huhns, M.N., Distributed Artificial Intelligence, Volume II. Pitman Publishers, Ltd., London, 1989.
22.Les Gasser, Ingemar Hulthage, Brian Leverich, John Lieb, and Ann Majchrzak. "Organizations as Complex, Dynamic Design Problems," in M. Filgueiras and L. Damas, eds. Progress in Artificial Intelligence. Lecture Notes in Artificial Intelligence 727, Springer Verlag, 1993.
23.Les Gasser and Toru Ishida. "A Dynamic Organizational Architecture for Adaptive Problem Solving," In Proceedings of the National Conference on AI, pages 52-58. July, 1991.
24.Anthony Giddens. The Constitution of Society, University of California Press, 1984.
25.Toru Ishida, Les Gasser and Makoto Yokoo. "Organization Self-Design of Distributed Production Systems." IEEE Transactions on Data and Knowledge Engineering, 4(2): 123--134, 1992.
26.Yan Jin and Raymond Levitt. "The Virtual Design Team: A Computational Model of Project Organizations." Computational and Mathematical Organization Theory, 2(3): 171---196, 1996.
27.Edward E. Lawler and John G. Rhode, Information and Control in Organizations, Goodyear Publishing Company, Pacific Palisades, CA 1976.
28.G. G. Lendaris. "On the Definition of Self-Organizing Systems", IEEE Proceedings 52, p. 324-325, 1964.
29.Thomas W. Malone. "Modeling Coordination in Organizations and Markets." Management Science, 33: 1317--1332, 1986.
30.Peter K. Manning, "Rules in an Organizational Context", in Organizational Analysis, Sage, 1977.

31. James G. March and Herbert A. Simon. Organizations. Wiley, 1958.
32. James G. March, Xueguang Zhou, and Martin Schulz. Dynamics Of Rules: Change In Written Organizational Codes. Stanford University Press, Stanford, CA. 2000.
33. M. V. Nagendra Prasad, Victor R Lesser and Susan E Lander. "Learning Organizational Roles in a Heterogeneous Multi-agent System" in Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multi-Agent Systems, 1996. http://citeseer.nj.nec.com/prasad96learning.html
34. G. Nicolis and I. Prigogine. Self-Organization in Non-Equilibrium Systems, Wiley, New York, 1985.
35. Brian T. Pentland. "Grammatical Models of Organizational Processes." Organization Science, 6, 5, pp. 541-556, 1995.
36. Brian T. Pentland and H. H. Reuter. "Organizational Routines as Grammars of Action" Administrative Science Quarterly 39, 3, pp. 484-510, 1994.
37. W. W. Powell and P. J. DiMaggio. The New Institutionalism in Organizational Analysis. The University of Chicago Press, Chicago, IL, 1991.
38. Michael J. Prietula, Kathleen M. Carley and Les Gasser, editors. Simulating Organizations: Computational Models of Institutions and Groups. AAAI Press/MIT Press, Cambridge, MA, 1998.
39. Gerald R. Salancik and Hussein Leblebici. Variety and Form in Organizing Transactions: a Generative Grammar of Organization. Research in the Sociology of Organizations, 6:1--31, 1988.
40. Scerri, P., Tambe, M., Lee, H., Pynadath, D., et al. "Don't Cancel My Barcelona Trip: Adjusting the Autonomy of Agent Proxies in Human Organizations." Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents: The Human in the Loop, 2000.
41. W. Richard Scott. Organizations: Rational, Natural, and Open Systems, 3rd Edition, Prentice Hall, Englewood Cliffs, NJ., 1992.
42. W. Richard Scott. Institutions and Organizations, Sage, 1995.
43. Herbert A. Simon. Administrative Behavior. Free Press, New York, 1947.
44. Herbert A. Simon. The Sciences of the Artificial, 3rd Edition, MIT Press, 1996.
45. Strauss, Anselm L. Continual Permutations of Action, Aldine de Gruyter, 1983.
46. Milind Tambe. "Agent Architectures for Flexible, Practical Teamwork." In Proceedings of the National Converence on Artificial Intelligence, AAAI Press, 1997.
47. James Thompson. Organizations in Action. McGraw-Hill, New York, NY, 1967.
48. Christoph von der Malsburg. "Network Self-Organization." In Steven F. Zornetzer, Joel L. Davis, and Clifford Lau, editors. An Introduction to Neural and Electronic Networks, chapter 22, pages 421--432. Academic Press, New York, 1990.
49. Heinz von Foerster. "On Self-Organizing Systems and their Environments", in: Self-Organizing Systems, Yovitts M.C. & Cameron S. (eds.), (Pergamon, New York), p. 31-50 1960. Well written, many interesting observations. Proof of meaningless of the term "SOS", first (?) discussion of "growth of phase space" route to organization, on relative information, order from noise principle.
50. Stanley Wasserman and Katherine Faust. Social Network Analysis: Methods and Applications. Cambridge: Cambridge University Press, 1994.
51. Karl E. Weick. The Social Psychology of Organizing, 2nd Edition, McGraw Hill Higher Education, 1979.
52. Wilson, D.S. & Sober, E. Reintroducing Group Selection to the Human Behavioral Sciences. Behavioral and Brain Sciences 17 (4) pp. 585-654, 1994. Also http://cogsci.soton.ac.uk/bbs/Archive/bbs.wilson.html (4/2001)

Multi-agent Infrastructure, Agent Discovery, Middle Agents for Web Services and Interoperation

Katia Sycara

Carnegie Mellon University
Pittsburgh, PA 15213
katia@cs.cmu.edu

Abstract. This chapter has two parts. In Part I, we present an overview of issues in modeling Multi Agent Systems (MAS), discuss what features and components are required for a MAS infrastructure, and present a model of a generic infrastructure. In addition, we present RETSINA as an example of an implemented MAS infrastructure. In Part II, we present issues in agent and service discovery and interoperation through a set of domain independent active and intelligent registries, called middle agents.

Introduction

Multi-Agent Systems (MASs) are becoming increasingly important: as a scientific discipline, as a software engineering paradigm, and as a commercially viable and innovative technology. Despite the considerable research that has gone into the formation of theories, scientific principles and guidelines for MASs, there is relatively little experience with the building, fielding and routine use of them. It is admittedly the case that the development of a MAS is extremely challenging, both in the laboratory but especially in the real world. However, MAS research will not fulfill its potential until we have a critical mass of fielded systems, components, and services. To achieve this goal, a stable, widely used, widely accessible and extensible MAS infrastructure is crucial. Various standards bodies (e.g. FIPA) are attempting to define standards for various aspects of MAS infrastructure, such as Agent Communications Languages. Various labs and companies are developing agent toolkits that could be reused for building agents and Multi-Agent systems. However, there is no coherent account of what constitutes a MAS infrastructure, what functionality it supports, what characteristics it should have in order to enable various value-added abilities, and what its possible relation with and requirements it may impose on the design and structure of single agents.

Another equally important aspect of MAS operating in an open world like the Internet, where communication links, information sources, services and agents can appear and disappear dynamically is the issue of *discovery* and *interoperation* of agents. White pages and yellow page registries of companies, for example have been proposed and implemented (e.g. Yahoo business categories) for human understandability. We have coined the term *middle agents* [12] to describe various agent intermediaries that can act as brokers and discovery services for agents on the

Internet. These domain independent intermediaries facilitate the finding and matching of agents and services with desirable functionalities (e.g. an agent that finds weather information).

Such intermediaries start having appeal for industry. For example, industrial organizations (e.g. SUN) are developing and making accessible software that could constitute a part of a MAS infrastructure, such as JINI for service discovery. Protocols such as UDDI (www.uddi.org), SOAP (www.soapware.org) and languages such as WSDL (www.wsdl.org) ebXML and e-speak are receiving increased visibility.

This chapter has two parts: The purpose of Part I (sections 1-3) is to acquaint the reader with important issues in modeling and development of Multi-Agent Systems (MAS). We present an abstract model of what a MAS infrastructure should be and present the Reusable Environment for Task Structured Intelligent Network Agents (RETSINA) Multi-Agent infrastructure as an instance of the general infrastructure model. Part II (sections 5-8) deals with agent discovery and interoperation through various domain independent middle agents.

The rest of the paper is organized as follows: in section 1 of Part I, MAS Infrastructure, we clearly define what we mean by MAS Infrastructure; in section 2 we define a general model of MAS infrastructure; in section 3, we present RETSINA as an implemented instance of the general infrastructure model; section 4 presents an introduction to the issues of agent discovery and interoperation through middle agents; section 5 presents the core functionality of middle agents; section 6 presents different types of middle agents, their distinguishing characteristics and protocols; section 7 presents different types of coordination provided by middle agents and discusses implemented systems that exhibit these types of coordination; section 8 presents conclusions and section 9 acknowledgments. Finally, we present an extended bibliography of relevant references.

PART I: MAS Infrastructure

1. Background

In this paper, we will distill our experience of recent years into an account of what constitutes MAS infrastructure, and specifically, what characteristics and abilities different parameters within the infrastructure afford. Our definition and treatment of MAS infrastructure will not be as encompassing as the one proposed in [20]. It will be concerned mainly with technology development, applications and use, rather than involving scientific and educational MAS activities[1]. This account of the MAS infrastructure has resulted from our vision that the *computational world will soon be populated with Multi-Agent societies* that are heterogeneous in agent structure, Multi-Agent organization and functionality. One important element that our account articulates is the relation between infrastructure for a *single agent* and the

[1] Of course having a technological infrastructure does positively impact those two activities also.

infrastructure for the MAS in which the agent participates. We consider MAS infrastructure to be the domain independent and reusable substratum on which MAS systems, services, components, live, communicate, interact and interoperate, while the single agent infrastructure is the generic parts of an agent that enable it to be part of a Multi-Agent society, i.e. to be socially aware.

2. Infrastructure Model

Agents in a MAS are expected to coordinate by exchanging services and information, to be able to follow complex negotiation protocols, to agree on commitments and to perform other socially complex operations. We define the infrastructure of a MAS as the set of services, conventions, and knowledge that support such complex social interactions. Agents need services to enable them to find each other in open environments, to communicate, to warrant that the proper security constraints are satisfied. Conventions, such as Agent Communication Languages (ACLs), and conversational policies are the basis for achieving interoperability and agreement on what the agents are doing and what they are achieving; knowledge of how to use the infrastructure, ACL and protocols as well as a common ontology is needed by the agents so that they can be effective participants in the community.

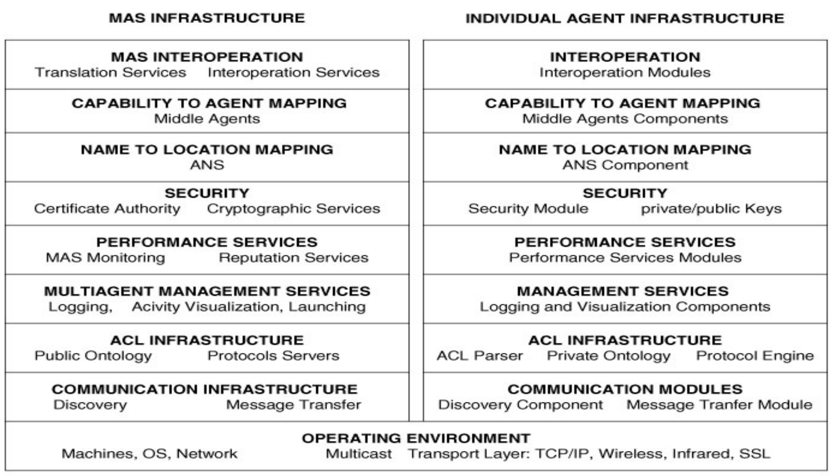


Fig. 1. MAS Infrastructure and Individual Agent Infrastructure that allows an agent to be part of a MAS

Crucially, the above definition does not mention what the infrastructure should know about the internals of the agents in the system. We claim that from the point of view of the MAS infrastructure, agents are "socially aware" programs that communicate, interact among themselves and with the infrastructure components, and whose behavior conforms to the rules of the MAS. An agent's problem-solving capabilities, however, are a black box to the infrastructure.

Figure 1 shows how the different services provided by a MAS infrastructure are organized in an abstraction hierarchy, in which the higher levels rely on the functionalities implemented by the lower levels. The infrastructure diagram has two parts: the MAS infrastructure, and the single agent infrastructure that allows an agent to be part of a MAS. The diagram also shows how the components of the infrastructure are reflected in the internal structure of an agent. In the diagram, the Problem Solving layer of an agent is absent precisely because the infrastructure does not make any assumptions about it.

Our claim has profound consequences: first it defines MASs as inherently heterogeneous, in the sense that any agent can enter the system and interact with the other agents independently of its internal architecture and model of the world. Similarly, the MAS infrastructure is mute on the points of particular coordination regimes. We claim that the MAS infrastructure should be general enough to support various coordination schemes such as team behavior [81], negotiation [73], Contract Nets [70] etc. This is why there is no coordination layer in the figure. In addition, we feel that social norms [8] are not part of the infrastructure but are particular to the design of a given MAS society.

In the following subsections, we provide a description of the infrastructure layers.

2.1 Operating Environment

At the bottom of the conceptual layering of the infrastructure, a MAS relies on an *Operating Environment*, i.e., on physical computers, on their operating system, on different types and topologies of the networks that connect different agents and different means of information transport. Single agents also use this infrastructure without any additional components or awareness. This is why the "operating environment" layer runs across both the MAS infrastructure and the single agent infrastructure portion of the figure. This level of abstraction should be totally transparent to the agents and the MAS, which should work across different platforms and networks.

2.2 Communication Infrastructure

A MAS is implemented on top of a *Communication Infrastructure* that transfers messages between the agents as well as between the agents and the MAS infrastructure. Current communication channels have various modalities, such as wired, wireless, infrared etc. To ensure maximum flexibility in MAS communications, the communication channel should support different modalities of communication between agents, such as synchronous or asynchronous communications, as well as be abstracted from the actual transport layer and the ACL used [62]. ACL independence does not mean that the ACL can be under-specified within a specific MAS, rather it means that the same communication infrastructure can be reused by different MASs that use different ACLs.

Independence from the transport layer guarantees that agents can communicate whenever there is an open connection between them, independent from the way in which this connection is implemented and from contingency situations that are not under the control of the agents. For example an agent should be able to be connected to other agents via a socket connection, or via infrared or with some sort of wireless radio connection. No matter what media is used, if there is an open connection between the agents, they should succeed in communicating.

Within an individual agent, a communication infrastructure is needed, i.e., an ACL-independent communication module that formulates an agent's messages, taking into consideration particular communication channel characteristics (e.g. wired, wireless).

Another important infrastructure service at this layer is the *discovery of infrastructure components*. For example, when an agent first comes up in an open environment, it may want to register itself with Agent Name Services (ANS). Instead of having hardwired IP addresses for such services, the MAS infrastructure and the corresponding single agent infrastructure can facilitate the discovery of existing ANSs. UPnP and JINI are examples of such discovery protocols. (See also section 3 for description of such infrastructure discovery protocols implemented in the RETSINA infrastructure).

2.3 ACL Infrastructure

An essential part of creating a community of agents is the specification of a language that can be spoken and understood by all the agents in that community. For this reason, the specification of an ACL, protocols and conversational policies used by the agents is an essential part of the specification of the MAS and it constitutes a part of the MAS infrastructure.

An ACL should specify the syntactic form of the messages exchanged. In addition, it should specify the semantic interpretation of the messages, so that an agent understands what the messages that it receives are all about. The interpretation of the messages relies on the specification of a shared ontology in which the terms used are defined. In turn, the ontology can be used to extract the meaning of the messages themselves. Conversational policies [26] and protocols embody the roles and social context [68] of agent communication. The social context constitutes the pragmatics against which agent communications are interpreted and used.

Correspondingly, an individual agent's infrastructure should support interpretation of a message by an agent, and facilities for allowing an agent to send messages. In addition, the agent should know what to do with the message it receives, i.e., how to parse the message, and how to interpret it in the context of an on-going conversation. Therefore, along with the ACL there should be a definition of a set of protocols [70] and conversational policies that specify what an agent's role is and how a message fits in the general scheme of the messages exchanged by the agents. For instance, a request for information should be followed by an answer or by a “sorry message”: an acknowledgment that the agent cannot provide an answer. In addition, an agent's language infrastructure should support the understanding of some public ontology that expresses the conversational content.

Most implemented research MASs, for example RETSINA [61] DECAF [25], Infosleuth [51], Jade [32] among others, use KQML [14] or FIPA ACL [16] as agent communication languages. Although, there are no industrial MAS infrastructures as such, XML is strongly favored to express message content in industrial systems.

OAA [46] agents exchange messages in the form of PROLOG predicates. One key difference between the ACL used by OAA and KQML or FIPA is that in the OAA ACL there are only two performatives: “solve” that is used to query other agents, and “solved” that is used to answer the query. But there is no way to express a performative equivalent to assertions like the “tell” in KQML. As a consequence, OAA agents are forced to maintain a precise history of the message exchange and infer from it what kind of message they received and what they should do with that message.

2.4 Multi-agent Management Services

MAS infrastructures should also provide additional system operation services, which we labeled *Multi-Agent Management Services* in Figure 1. Such services provide facilities that support the work of a MAS over time: they include *Logging* facilities that record the messaging activity of agents in the MAS*; Management Tools* that monitor and visualize the activity of the MAS; and *Installation Services* and *Launching Services* that ease the burden of starting and configuring the many agents that comprise a MAS.

Starting many agents on multiple platforms at the same time is a very time consuming process. In RETSINA we developed a launching and management system for our agents. This system is in charge of starting agents on different machines in the local network. RETSINA also provides tools for monitoring the activity within the MAS and also management facilities.

A similar system is used by ZEUS [52], which implements a visual editing system that allows the programmer to construct the MAS and to specify the interactions between the agents. The editing system can also be used for monitoring and management facilities. OAA implements an application called “startit” that starts, manages and shuts down the system.

2.5 Performance Measurement

Because MASs are in general heterogeneous, the agents differ in ability, efficiency, reliability, etc. The MAS should provide *Performance Measurement* to monitor the performance of the agents. For example Performance Measurement services could be used to optimize the distribution of tasks across agents. Such services could rank MAS services in terms of performance, so that the more efficient would be more likely to receive requests. Also, the reputation of agents might be monitored [89]. Any agent that provides false or unreliable information would lose credibility within the MAS and it would not be used by any agent that needs its
service. In addition, failures could be monitored and the information collected could be used for failure tracking or facilitating failure recovery.

Although the performance measurement services for MAS could operate without the individual agents being aware of them, there could be corresponding services within an individual agent that increase agent effectiveness as a MAS participant. For example, an agent could be self-aware, i.e., monitor its own performance and try to optimize it. Or, an agent could monitor its own failures and try to recover from them.

2.6 Security

Agents in an Open MAS, where agents can join and leave the society dynamically and where agents have been designed by different development groups, meet as perfect strangers. Each agent knows very little or nothing about the agents with whom it interacts. Therefore, security services are needed to ensure that agents do not misbehave.[2]

The security layer of the MAS infrastructure deals with these problems. It defines a set of trusted services, as for example certificate authorities, that guarantee the identity of the agents, and a set of protocols that are guaranteed to prevent voluntary and involuntary losses of goods, services, or other values during the interaction.

Individual agent infrastructure should make sure that agents in the system can interact with these Security services. Such an example would be an agent interacting with the Certificate Authority to retrieve the keys necessary to perform its transactions. Furthermore, agents should know and be able to handle encryption and to follow the secure protocols.

Security is a concern in MAS implementations because, as we discussed above, agents can misbehave by cheating on other agents or by affecting the integrity of the system [45; 29]. Yenta [17] as well as RETSINA implements a security system to protect the integrity of its Matchmaker. Security is a major concern in the mobile agents community [27], since agents have access to a remote host and their misbehavior might damage the host as well as the MAS infrastructure the agent belongs to.

2.7 ANS: Mapping Names to Agent Locations

A MAS infrastructure includes facilities to find agents by some identifying feature, such as a name. MAS can be divided in two classes: systems that abstract from the physical location of agents and systems that do not. CGI-BIN scripts on the Web are an example of a MAS that employs fixed locations. Each CGI-BIN script is addressed by the name of the web server on which it is running and the exact location within such a server. When the CGI-BIN script is moved to another location, all the references to it should also be updated, but there is no provision in the HTTP protocol or anywhere else that does it automatically. Furthermore, while new CGI-BIN scripts are constantly added and removed from the web, any reference to them is hardwired either in a HTML form or in other CGI-BIN scripts, since there is no mechanism nor provision that allows web pages to reconfigure automatically to make use of new services provided nor to detect when services that they used to access are no longer available.

[2] Most common security issues include communication security and infrastructure integrity. Communication security guarantees that a message cannot be eavesdropped, authentication, so that the agents cannot spoof each other, and non-repudiation i.e.: disallow agents to deny having taken part in a transaction. Infrastructure integrity guarantees that no agent can manipulate the information stored in the infrastructure components such as the ANS and the Matchmaker. In addition, Communication Integrity guarantees that an unauthorized agent cannot change the contents of a message.

In the general case, agents can join and leave a MAS dynamically and unpredictably. Agents that are not bound to a particular physical location can appear anywhere on the net and still be part of the community of agents. While this flexibility provides an essential advantage because an agent developer does not need to care where an agent is located, it requires services for mapping the agent name dynamically to the agent location. In addition, such facility provides the basis for agent mobility. No agent that is bound to a precise location can move and still be part of the MAS. To abstract from the physical location of the agent, the MAS infrastructure should maintain a registry to map the name of the agent to a physical location so that it can eventually be reached. Such a registry is represented in Figure 1 as the ANS: Agent Name Server. An ANS is like a DNS but with increased flexibility for real time updates, discovery services, automatically ``pushing" agent name registration to other ANSs, etc. Systems that are based on the CORBA ORB [11] such as the Sensible Agent Testbed [4], or on JINI [36], such as the Grid [10], or on the RETSINA ANS infrastructure [41] (see section 3.7) use an underlying infrastructure that automatically abstracts from the physical location of the agents.

The *ANS Component* in the figure is the corresponding individual agent infrastructure that registers and un-registers with the ANS and initiates lookup requests for a desired agent.

2.8 Mapping Capabilities to Agents

A general MAS should support an open rather than closed agent world.[3] Open systems allow agents to enter, and exit, the system dynamically and unpredictably, while closed systems employ a fixed set of agents that are known a priori. Since in an Open MAS the set of agents is not known a priori, the infrastructure should provide ways for its agents to locate each other based not only on name but also on functionality or capability. Locating agents by capability is solved by employing a set of infrastructure agents called *Middle Agents* [12]. Some examples of middle agents reported in the literature include the OAA Facilitator [46], the RETSINA Matchmaker [77; 78] and the Infosleuth Broker [58]. Middle Agents maintain an up-to-date registry of agents that have made themselves known to the MAS community, along with the services that each agent provides. This
information is called the agent's capability *advertisement* and is provided by the agent to a middle agent. When an agent needs another that has some required capability, it sends a middle agent a *request* specifying the desired capability. The middle agent matches requests and advertisements. In general, there could be a variety of middle agents that exhibit different matching behaviors and have different performance characteristics. In prior research, we have identified 28 middle agent types and have experimented with different performance characteristics, such as load balancing, fault tolerance etc. [12; 86].

Whether the system allows Middle Agents or not affects the infrastructural requirements of a single agent. An agent must have the ability to construct

[3] In a closed MAS, each agent knows the name, location and capability of the others. Thus agent interactions can be statically predefined. This makes agent design and construction simple, but makes the MAS brittle and not extensible.

advertisements to make itself known to the agent community and also construct requests to take advantage of services provided by other agents. If an agent lacks these abilities, it would be stand alone and isolated from the MAS activities. In Part II of this chapter, we focus specifically on the Discovery and Brokering of Information, Agents and Services on the Internet.

2.9 Interoperation

It is clear that as the number of MAS created by different groups increases, there will be an increased need for MAS interoperation. The development of sharable ontologies, conversational policies, ACLs and translation services will go a long way towards allowing individual agents to interoperate. However, additional infrastructure is needed to take care of MAS architectural mismatches, for example between a centrally controlled MAS that uses a Facilitator as middle agent and a distributedly controlled MAS that uses a Matchmaker as middle agent. Each MAS may have its own architecture-specific features, such as: agent registration, agent capability advertisement, agent communication language, agent dialogue mediation, default agent query preference, and agent content language. Since MAS are in general open, there is the further requirement that interoperation must be done in real-time so as to capture the dynamics of the agent world. If an agent enters one MAS community, agents in the other MAS communities should have ways of finding and transacting with this agent, if it matches a required capability.

Currently, only a couple of research interoperation systems exist (see sections 3.9) between architecturally different Open MASs. We believe this area will receive increased attention, as more MASs get developed and deployed.

3. The RETSINA MAS Infrastructure

Over the past few years, the Intelligent Agents Group at Carnegie Mellon University[4] has developed a Multi-Agent infrastructure, the Reusable Environment for Task Structured Intelligent Networked Agents (RETSINA). The Group has had a long history in researching various issues in MAS, such as MAS stability [82], MAS learning [1], MAS coordination [42]. In addition, we have been building and experimenting with MAS over several years [74; 79].

We made various design decisions that were motivated by our assumptions of what is the best added value that future MAS could provide. In this chapter, we will describe the RETSINA infrastructure as an implemented instantiation of the proposed abstract infrastructure model and point out the particular design decisions and

[4] More information on the activity of the Intelligent Agents Group can be found at http://www.cs.cmu.edu/~softagents. The infrastructure part of this chapter is joint work with Brent Langley, Massimo Paolucci, Joe Giampapa and Martin Van Velsen of the Intelligent Agents Group at CMU. Part II of this paper is joint work with Matthias Klusch of DFKI.

characteristics it embodies. The RETSINA infrastructure has evolved over the years. We have used it to implement a variety of applications in order to test the generic features of the infrastructure to make sure of its generality. Such applications include agent-based e-commerce [83], negotiation [73; 90], agent-based coalitions for volume discounts [87], in-context information gathering, financial portfolio management [75], agent-based aircraft maintenance [64], and others. Each subsequent application guided the refinement of the infrastructure towards increased generality and flexibility.

RETSINA is an open MAS infrastructure that supports communities of heterogeneous agents. The RETSINA system has been implemented on the idea that agents in the system should form a community of peers that engage in peer-to-peer relations. Any coordination structure in the community of agents should emerge from the relation between the agents rather than being imposed by the infrastructure. Following this premise, RETSINA does not employ any centralized control on the MAS, rather it implements distributed infrastructural services that facilitate the relations between the agents instead of managing them.

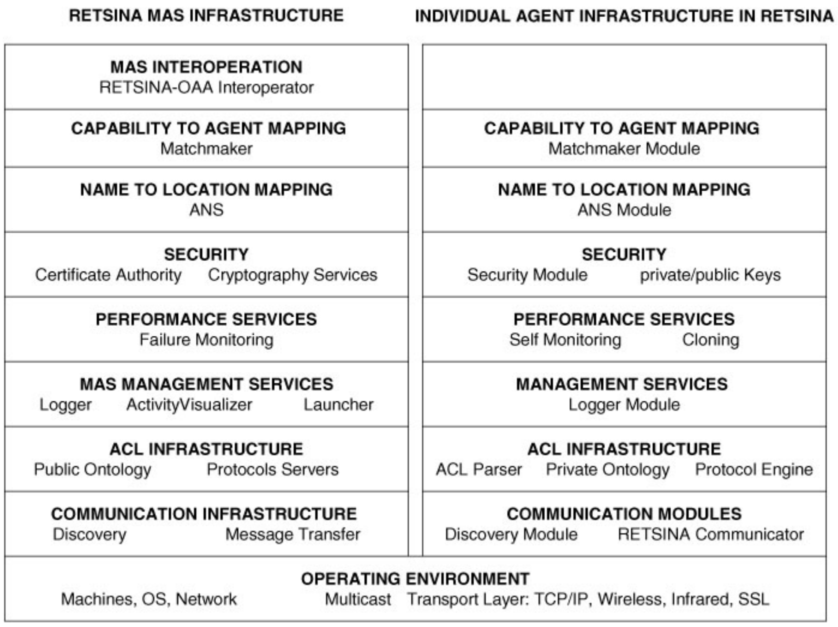


Fig. 2. The RETSINA MAS Infrastructure and Individual Agent Infrastructure

The organization of the RETSINA MAS infrastructure is displayed in Figure 2. It shows how the various components are organized on the basis of the infrastructure model in Figure 1. In the rest of this section we will describe these components.

3.1 Operating Environment

The RETSINA MAS is independent of the platform on which the infrastructure components and the agents run, and it automatically handles different types of transport layer.

Applications of the RETSINA MAS are routinely distributed on different platforms ranging from different versions of Windows, different versions of Linux, and Sun OS. Furthermore, they include agents running on PalmPilots. The agents used have been implemented in different languages as Java, C, C++, Python, LISP, and Pearl. Communication transport layers handled by RETSINA include TCP/IP, wireless, SSL, infrared, and serial connection.

3.2 Communication Infrastructure

RETSINA is based on two types of communication channels: one provides message transfer for direct peer to peer communication between the agents, the other is based on multicast used for a Discovery process that lets the agents find infrastructure components.

Direct message transfer is supported in an individual agent by the RETSINA communicator [62] that provides an abstraction over the physical transmission layer abstracting over the type of network used. The Communicator supports synchronous as well as asynchronous communication, and it manages multithreaded communication that allows the agent to maintain conversations with multiple agents at the same time.

3.2 Discovery Protocols

Discovery uses multicast to connect agents to the infrastructure components. For example, an ANS announces its presence by multicasting. An agent can also announce its presence by multicasting a request for an ANS. If the agent finds an ANS, it registers with it, or performs a lookup request. To reduce the load on the multicast channel, no negotiation happens directly on multicast; direct transactions between the agents and the infrastructure are performed on a direct channel.

The use of Discovery allows flexible entrance of agents and infrastructure in the RETSINA MAS [41]. An agent can enter the MAS when no infrastructure is yet present, and wait until infrastructure components enter the system. After these components multicast their presence, the agent registers with them and from that moment on it is effectively a reliable resource for the agent community.

Discovery is also very useful for agents running on mobile platforms. While these agents might not be mobile themselves [18] they may move around just because the platform they are running on is moved. Using Discovery in the new place, the agent can re-orient itself and find the local components of the infrastructure.

In RETSINA we use discovery as a first point of contact of an agent with a running Multi-Agent system. It is also a fallback mechanism in case one or more infrastructure agents fail (ANS, Matchmaker, Etc). Here discovery is used to located other available infrastructure agents.

3.3 ACL Infrastructure

The ACL used in the RETSINA MAS is KQML [14]. Messages exchanged by the agents have two components: one is the specification of the content of the message, the other is an envelope that specifies information such as sender, receiver, thread of conversation, ontology and language used in the content part. The RETSINA infrastructure dictates the format of the envelope, because it is used to deliver the message, but it does not make any assumption on the content of the message itself. Any content would do as long as the agent that receives the message can understand it.

The specification of the language does not guarantee that the agents understand each other. They also need to have a shared vocabulary and shared concepts that specify the meaning of the words that the agents use. For this reason RETSINA provides an ontology based on diverse domain-specific taxonomies of concepts derived from the Wordnet [13].

The RETSINA MAS also provides a protocol engine and a protocol language that allows agents to specify their roles and the messages to be exchanged and expected in the context of a protocol. The protocols employ social semantics [68]. Currently, the specification and implementation of the protocols is in the form of finite I/O automata.

3.4 MAS Management Services

The RETSINA MAS includes three management components: the *Logger, ActivityVisualizer* and *Launcher* that form an initial set of tools that help with monitoring, debugging and launching MAS applications.

The *Logger* records the activity of the agents. Specifically, the Logger records agent entry to and exit from the system, and the exchange of messages.

The Logger is connected to the *ActivityVisualizer* that displays the activity within the system. The ActivityVisualizer uses the information provided by the Logger to display in real time, which agents are in the system, and their interactions. Furthermore, the Logger and the ActivityVisualizer can be used in playback mode thus permitting the agent programmers to review and analyze activity in the MAS.

The *Launcher* automatically configures and starts both infrastructure components and agents on different machines, platforms and operating systems from a single point of control, greatly reducing the work that has to be done by hand to start and maintain a distributed application. The launcher is of great value especially as different agent versions get developed and as agents may change resource requirements or as they need to be moved and restarted on different machines.

3.5 Performance Services

The current version of the RETSINA MAS does not include MAS performance service or reputation service. We have however built performance service monitors in simulation as well as distributed check-pointing and rollback upon agent failure. We have experimented with different monitoring services in the context of contract net family of protocol [70].

3.6 Security

Since RETSINA is an open system, unknown and possibly untrustworthy agents can enter at any time. These agents can damage the system in many ways: they can spy on other agents, steal goods or information, and damage the content of the infrastructure components. For instance, a malicious agent might prevent the MAS from working by un-registering all the agents from an ANS. The security infrastructure of the RETSINA MAS [85] prevents such problems from happening.

In RETSINA, we guarantee three types of security: agent authentication via a Certificate Authority, communication security, which guarantees that the communication between agents cannot be eavesdropped, and integrity of the components that guarantees that no component can be inappropriately manipulated [86]. Communication security is achieved by giving agents unique IDs, as private keys, which are verified using public keys, and by layering SSL underneath the communication interface used by the agents. Integrity of the MAS components, such as the ANS, is also guaranteed by relying on the unique IDs of agents and by adding access control mechanisms.

The security components of the RETSINA infrastructure for the individual agent are the Security Module in the agent and the Certificate Authority in the MAS infrastructure. The Security Module generates the private and public keys of the agent and it requests certification of the public key from the Certification Authority, which binds the requester's ID to its public key.

3.7 RETSINA ANS

In RETSINA, an ANS does not participate in the transaction between agents; it only provides them with addresses that they can cache removing the need for unnecessary lookups. In addition an ANS provides robustness of agent communication in the event of an ANS failure, since the agents can continue their transaction even when no ANS is available in the system.

Furthermore, multiple ANSs can be present in the system at the same time [35]. ANS servers find each other through Discovery using multicast within a LAN. Since it is not feasible to use multicast outside a LAN, RETSINA uses “reference ANSs” that are visible outside the local subnetwork boundaries. Through reference ANSs the lookup search for an agent can be spread to a much wider network and possibly the whole Internet [41].

Within an individual agent, the ANS component enables the agent to register and un-register with an ANS and request lookups of desired agents.

3.8 Middle Agents

Agents enter a MAS to exchange services with other agents, but since RETSINA is an open system, no agent can be sure of what services are available in the MAS at any given time, and who provides them. It is a task of the infrastructure to provide a registry of services available in the system and to allow agents to search for them in this registry.

RETSINA solves the service location problem by using a set of middle agents called Matchmakers distributed across the MAS [35]. Each Matchmaker records a mapping between agents in the system and the services that they provide. A Matchmaker uses two types of data: the advertisements of the services provided, and the requests from agents that need a service, both of them expressed in the LARKS [77; 78] language. The task of a Matchmaker is to find which advertisements match the requests. To accomplish this task a RETSINA Matchmaker uses the LARKS matching engine that performs both syntactic and semantic analysis of the advertisements and requests to find exact or partial matches.

RETSINA and DECAF [25] implement Matchmakers and ANSs as lookup services: the Matchmaker maps capabilities into agents; the ANS maps agents to locations. OAA [46] and InfoSleuth [51] implement brokers that map capabilities to agents. This mapping also contains information on the location of the agents. The first difference between RETSINA and DECAF on one hand and OAA and InfoSleuth on the other is that the first two implement a distributed control in which the matchmaker does not manage the interaction between the agents, while both the OAA Facilitator and the InfoSleuth Broker do. The distribution of services implemented by RETSINA and DECAF increases the reliability of the system. Furthermore, advertisements in RETSINA and DECAF [25] represent the functionalities of an agent by specifying the types of inputs that it requires and the types of outputs it generates. In contrast, the advertisement of an OAA agent is just predicates representing a sample query: it does not specify what information the agent requires to compute an answer or what information it returns. Finally, the advertisement in InfoSleuth is a classification of the agent in an ontology; it specifies what the agent is about, instead of what the agent does.

3.9 RETSINA-OAA Interoperator

As the number of autonomous agents and Multi-Agent systems (MAS) increases a problem of intra-MAS interoperation emerges. MAS developers in the same group usually make strong assumptions on the agent architecture and agent ACL. None of these can be assumed when different developers from different groups construct MASs without a strict coordination. Nevertheless, we would like agents from different MAS to interoperate and interact exchanging services and information.

Imagine an OAA agent trying to enter the RETSINA MAS. Such an agent would be totally lost and unable to interact with either the agents or with the infrastructure components. It would not be able to communicate with any agent because it would "speak" the Prolog-based OAA ICL, while every agent in the RETSINA system "speaks" KQML. Furthermore, it would expect to deal with a Facilitator, but may end up dealing with a Matchmaker instead, with the result that it would not be able to ask for services nor to interpret what is returned by the middle agent.

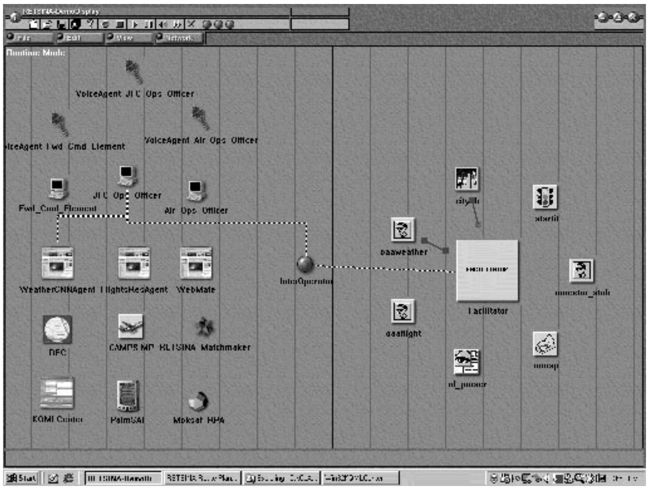


Fig. 3. The RETSINA-OAA InterOperator mediates between the RETSINA MAS (on the left) and the OAA MAS (on the right)

While many claims have been made about openness of MASs, the current practice is that MAS developers make such strong assumptions on the agents they develop that natural interoperation across MAS boundaries is virtually impossible.[5] To interoperate between OAA and RETSINA, we implemented the RETSINA-OAA InterOperator [23]. The task of the InterOperator is to allow any agent in the RETSINA system to access any service or information provided by OAA agents, and for any agent in the OAA system to access services or information provided by RETSINA agents.

The RETSINA-OAA InterOperator ``bridges" the two worlds of RETSINA and OAA by performing two types of tasks: first it makes the two systems visible across MAS boundaries; second it allows agents to exchange messages across MAS. The first task is accomplished by collecting all the advertisements of RETSINA agents, translating, and registering them with the OAA Facilitator. Similarly, the advertisements of OAA agents with the Facilitator are collected and advertised with the RETSINA Matchmaker. Therefore, through the RETSINA-OAA InterOperator, the two systems are able to “see” each other's agents. The second task is accomplished by translating the queries of the agents of one MAS to the agents of the other MAS, and then translating the answers back.

[5] Attempts at standardizations such as FIPA (FIPA, 2000) are likely to reduce the problem, but not solve it. Differences will remain in the Ontologies used, the interaction protocols and the MAS architecture.

Due to fundamental differences in the architectures and ACLs of the RETSINA and OAA Multi-Agent system architectures, it is not possible for all forms of agent-to-agent interaction of one MAS architecture, to be translated to the other. Nevertheless, the RETSINA-OAA InterOperator does adequately allow for the necessary agent interactions to occur across MAS boundaries.

PART II: Agent Discovery and Interoperation through Middle Agents

4. Introduction

Like intermediaries in the physical economy, intelligent middle-agents can be considered as electronic intermediaries in the digital economy. These agents provide means of meaningfully coordinating activities among agent providers and requesters of services (information, goods, or expertise) in the Internet [12]. Their main task is to locate and connect the ultimate service providers with the ultimate requesters in open environments, that is, to appropriately cope with the connection problem.

Various distributed and centralized settings exist to solve this problem. In general, we roughly distinguish three agent categories, *service providers* (providers), *service requester* (requesters), and *middle-agents*. The basic process of capability-based mediation by middle-agents has the following form: (1) providers advertise their capabilities to middle-agents, (2) middle-agents store these advertisements, (3) a requester asks some middle-agent to locate and connect to providers with desired capabilities which may include complete transaction intermediation and other value-added services, and (4) the respective middle-agent processes this request against its knowledge on capabilities of registered providers and returns the result. The result may be either a subset of the stored advertisements with names of respective providers to contact, or the result of the complete transaction for the most suitable service.

While this process at first glance seems very simple, it is complicated by the fact that the Internet, considered as an open social, information, and business environment, is in a continual state of change. Dynamic changes, concern, for example, location and heterogeneity of available resources and content as well as different user communities and societies of agents each of which is pursuing its goals that may conflict with the goals of others. In addition, the number of different agents and Multi-Agent systems that are being developed by different groups and organizations significantly exacerbates the connection problem.

Thus, the essential capabilities of any kind of middle-agents are to facilitate the interoperability of appropriate services, agents and systems, building trust, confidence and security in a flexible manner, and to comply with regulatory and legal frameworks when available. This depends on the specific requirements implied by given specific application, information, and system environment. Given the fact that different types of middle agents provide different performance trade-offs what types of middle-agents are appropriate depends on the application. The overall challenge of a middle-agent based solution is to fit in with the considered situation most effectively, efficiently and reliably.

5. Middle Agents

The notions of middle-agents, such as matchmakers, brokers, facilitators, and mediators have been used but not clearly defined. In the following, we address this terminological issue. In general, middle-agents are agents that help others to locate and connect to agent providers of services [12; 86]. Furthermore, any middle-agent can be characterized by its core functionality of (1) providing basic mediation services to the agent society, (2) coordinating these services according to given protocols, conventions, and policies, and (3) ensuring reliable service mediation in terms of leveled quality of services as well as trust management within and across Multi-Agent systems borders.

5.1 Service Intermediation

In open information and trading environments, such as the Internet a middle-agent has to provide basic mediation services for (1) the processing of agent capability and service descriptions, (2) semantic interoperation between agents and systems, (3) management of data and knowledge, and (4) distributed query processing and transactions. We will briefly discuss each of these service types.

*Processing of agent capability and service descriptions***.** Basic requirement to understand and automatically process requests for and descriptions of services and capabilities of agents, i.e. service profiles, is the agreed use of a (set of) common agent capability description language such as LARKS [77; 78], CDL [9], or XML-based service description frameworks, like RDF (S) [60] or eCo system's server and common business library (CBL) of generic XML document models for e-commerce [24].

In general, the middle-agent in real time must parse, validate, understand and respectively process capability and service descriptions it receives. This is in order to efficiently determine which of the advertised services and capabilities of currently registered service provider agents are most appropriate for a given request of a requester agent. The choice of a suitable matching mechanism certainly depends on the structure and semantics of the descriptions to be matched as well as on the desired kind and quality of the outcome of the matching process; it may rely, for example, on simple keyword and value matching, use of data structure and type inferences, and/or the use of rather complex reasoning mechanisms such as concept subsumption and finite constraint matching. Semantically meaningful matching requires the matching service to be strongly interrelated particularly with the class of services enabling semantic interoperation between agents.

Semantic interoperation. One main obstacle to the meaningful interoperation and mediation of services is the syntactic and semantic heterogeneity of data and knowledge the middle-agent does access and receive from multiple heterogeneous agents, information systems and sources. The functional capability of a middle-agent to resolve such structural and semantic heterogeneities refers to the knowledge-based process of semantic brokering. Most methods to resolve semantic heterogeneities rely on using partial or global ontological knowledge, which may be shared among the agents. This requires a middle-agent to provide some kind of ontology services for statically or dynamically creating, loading, managing, and appropriately using given domain-specific or common-sense ontologies as well as inter-ontology relations when it processes requests and data from different agents. For example, the agreed use of some common metadata catalogue and XML-based frameworks like Dublin Core and RDF(S), respectively, a standardized ontology exchange language (OEL) such as XOL or OIL [53], or the Darpa Agent Markup Language (DAML) (www.daml.org) could be used by different agents in a Multi-Agent system to represent and exchange individual ontological knowledge of given domains.

Management of data and knowledge. Basic internal services of a middle-agent include the efficient storage and management of data and knowledge about itself and other agents. Such data are, for example, sets of capability advertisements of provider agents, past and current requests of requester agents, ontological knowledge shared among agents and other auxiliary data for internal processing. The corresponding databases, repositories and knowledge base of a middle-agent may be browsed by other agents and users for different purposes such as, for example, the specification of appropriate requests in the domain or monitoring activities according to given valid access restrictions and security policies.

Distributed query processing and transactions. In some cases the middle-agent can be used to put forward queries to multiple relevant external databases or other sources to gather information on behalf of its requesters. This requires the middle-agent to perform distributed query planning and execution of transactions in collaboration with respective systems and agents.

5.2 Service Coordination

The coordination of mediation by a middle agent requires in particular its ability to meaningfully communicate with agents, systems, and users which are involved in the overall mediation process according to given protocols, conventions, and policies. As a basic additional means agent naming and registration is needed to enable the middle-agent to locate and identify its providers and requesters within and across multiple agent societies.

*Agent registration and naming***.** Location of relevant agents by the middle-agent presumes that its clients are registered and can be named at any instant. This implies utilization of basic services of agent registration and naming for example agent name servers (ANSs) within the network domain covered by the middle-agent.

Inter-agent communication. Any coordinated interaction between the middle-agent, registered provider or requester agents and other middle-agents basically relies on the agreed use of one or more standardized agent communication languages (ACL), given conversation protocols, and policies. In communicating information among different agents it is essential to preserve the desired semantics of utterances transmitted via messages of an ACL. Both the intention of the message as well as its data content has to be understood correctly by the middle-agent and its clients to perform and support required service mediation, respectively.

Accessing sources of data and information. If the middle-agent has to access database systems, knowledge bases or other sources of information by itself it can do so either via standardized APIs or transparent remote access methods such as JDBC/ODBC, OKBC, Java RMI, and CORBA, respectively.

In a different approach the middle-agent may retrieve the required data in collaboration with so-called wrapper agents, which are encapsulating the relevant systems and sources. The middle-agent forwards appropriate subqueries to the wrappers each of which transforms the received request to a query in the proprietary query language of the particular system and returns the result of its execution in the desired format to the middle agent. The middle agent then has to merge all of these partial results for further processing. Such a scenario is in compliance with the paradigm of a mediator for intelligent information systems introduced by Wiederhold in 1992 [84].

Interfacing with users. In addition to the requirement of a middle-agent to appropriately communicate with its clients and relevant systems, it may also provide interface services to human users to let them, for example, browse available information they are interested in such as parts of a local domain or shared global ontology, sets of actual service descriptions, or registered service providers. These interface services have to be restricted according to given security policies and requirements. For example, the RETSINA A-Match agent (www.cs.cmu.edu/~softagents/a-match) provides an interface so that human users can input agent capability descriptions and requests.

Execution of mediation protocols and policies. Any activities related to service mediation within an agent society can be organized and coordinated according to given protocols, conventions, and policies of interaction between agents. The middle-agent by definition plays a central role of initializing and coordinating the interaction among agents in collaborative or competitive settings. Most common forms of mediation protocols include matchmaking, brokering, and arbitration in negotiations between provider and requester agents, for example, at virtual marketplaces or auctions. Recent industrial protocols include UDDI (www.uddi.org) and SOAP (www.soapware.org).

5.3 Security and Reliability of Mediation

Each client requires the contacted middle-agent to be reliable in terms of trust and quality of service. That means that the middle-agent should perform its designated mediation services at any given instant in a trustworthy manner and in compliance with given security and quality policies of individual agents or the considered agent society as a whole. This particularly requires the middle-agent to cope with different kind of policies in a coherent and consistent manner.

Quality of service. A middle-agent has to assure that the data it stores, processes and provides to the agents in the society comply with desired data quality requirements, standards, and policies. In this context data policy is the overall intention and direction of an agent with respect to issues concerning the quality of data products and services it will receive either from the middle-agent or the respective provider agents. The specification, management and evaluation of quality of mediated data and services may follow corresponding methods, standards and metrics used for data warehouse quality management and software implementation quality such as the ISO 9126 standard.

Trust management. Besides the client-sided demand to the middle-agent of providing quality of services there is also the essential requirement of a middle-agent to behave in a trustworthy manner to its clients in terms of guaranteeing desired data privacy, anonymity, and verification of claimed agent capabilities [86]. In this sense, a middle-agent acts as a trusted intermediary among the agents of the considered agent society, according to given trust policies of individual agents and the whole society.

Both internal data and knowledge of, and the computational processes of mediation executed by the middle-agent should be robust against external manipulation or attacks of malicious agents, systems, or users. On the other hand, clients of a middle-agent should have no incentive to misuse any information, which has been revealed by the middle-agent during mediation. In this respect most common trust actions are authorization and verification of credentials of all parties, which are involved in the mediation.

In summary, there is an obvious need for all agents involved in the mediation process to use an appropriate trust model to analyze and assess the risks of and methods to prevent and counteract attacks against their data and knowledge. Models, methods and techniques supporting the establishment and management of mutual trust between a middle-agent and its clients in an open environment include

- the use of expressive trust establishment certificates useful for, e.g., binding agent identity to public key infrastructures such as IETF's SPKI, and other standard security mechanisms and protocols, the use of mechanisms to bind agent names to their human deployers, so that the human would bear responsibility in case his/her agent misbehaves.
- the formal specification of agent trust policies, and
- the respective update, propagation, and transitive merge of trust matrices to calculate an overall trust relationship that accounts for the trust values in each and every individual trust relationship which is relevant to the considered mediation process. An option is the application of distributed history-based reputation mechanisms to agent societies.

6. Types of Middle Agents

Decker et al. [12] investigated different roles of middle-agents in the solution space of the connection problem from the standpoint of privacy considerations. The authors particularly examined knowledge about requester agent preferences, and provider agent capabilities; in this view specific requests and replies or actions in service of a request are interpreted as instances of preferences of requester agents and capabilities of provider agents, respectively. Both preference and capability information can initially be kept private at the requester agent, be revealed to some middle-agent, or be known by the provider agent.

For example, a broker agent understands but protects the initial privacy of capability and preference information of both the requester agent and provider agent; neither the requester agent nor the provider agent ever knows directly about the other in a transaction, which is intermediated by the broker agent. In contrast, any requester agent can query capability information of registered provider agents at a matchmaker or yellow-page server such that this information is revealed initially to both requesters and providers.

In [86], a broader classification of middle agents was made along the following dimensions: (1) Who sends initial information to a middle agent, a provider or a requester? (2) Is the information sent only capabilities (requests) or does it also specify preferences? (3) How is the information received by a middle agent communicated? (4) Is the content of a middle agent database browsed or queried? (5) How much information is specified in a query to a middle agent? (6) Does the middle agent intermediate transactions?

Given that each dimension only allows for two possible answers, we use an identification scheme where different types of middle agents are identified by binary numbers of 6 digits: given a middle agent, its type identifier is a number whose nth digit reflects the value of its nth dimension (0 or 1). Considering only valid combinations of binary values in the 6 dimensions, we get 28 different types of middle agents. For example, middle agents [001100], [011100] and [011110] are variants of matchmakers. Wong and Sycara [86] gave a formal specification of the respective protocols by means of state-transition relations based on input/output (IO) automata [44]. An IO automaton performs an action in a given state such that it may transition to a new state according to a given transition relation. This relation is described in a precondition-action-effect style to specify the changes that occur as a result of an action in the form of a simple pseudo-code that is applied to a pre-state to yield another state. Each agent is modeled as a process, which, in turn, is specified as an IO automaton.

Different types of middle agents exhibit different performance characteristics in terms of privacy, robustness, adaptivity, etc. [12] So, the use of particular type of a middle agent depends on the requirements of the application. Providing a taxonomy of middle agents is a first step towards standardization of middle agents and their protocols of interaction.

The coordination of mediation activities within and across agent societies can be performed by a middle-agent, for example, via brokering or matchmaking. Both types of mediation imply different requirements and protocols for interaction among agents, which are involved in it. In addition, ontology services are typically used to perform meaningful automated reasoning on capability and service descriptions specified in a given Agent Capability Description Language (ACDL).

In the following we briefly summarize the high-level interaction patterns for both the matchmaking and the brokering protocols.

6.1 Facilitator/ Broker

A broker agent may actively interface requester agents to provider agents by intermediating requested service transactions. All communication between paired requester and provider agents has to go through the broker. It typically contacts (a set of) the most relevant provider agent, negotiates for, executes and controls appropriate transactions, and returns the result of the services to the requester agent.

Broker agents typically do not provide a global, semantically integrated, consistent information model to their clients but store collected information together with associated ontological annotations in some standardized data (structure) format in one (or multiple) appropriate repositories like in a data warehouse. Brokers may subscribe to certain provider agents if needed. The interaction of brokers with other agents is neither restricted to wrappers or mediators nor committed to a fixed number of agents. This ability is particularly useful since brokers are usually acting in dynamic environments in which resources and agents may continually enter and leave the agent society. In such environments it turns out to be inappropriate to maintain a static pre-integrated global model that is valid for a rather closed society of agents and systems.

Based on the functional capability of broker agents the interaction pattern of brokering is shown in the figure below.

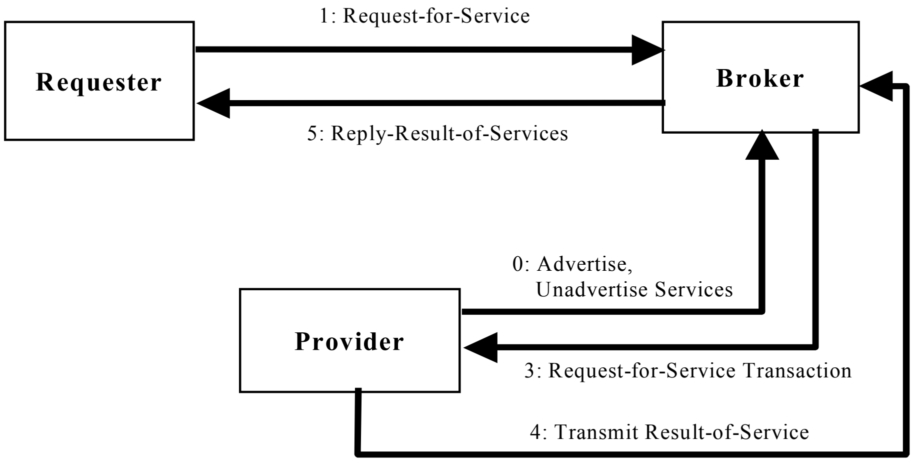

1: Request-for-Service
Requester
Broker
5: Reply-Result-of-Services
0: Advertise, Unadvertise Services
Provider
3: Request-for-Service Transaction
4: Transmit Result-of-Service

Fig.4. Interaction pattern of capability and service brokering between agents

However, it is noteworthy that the semantics of the terms matchmaker and broker as well as mediator and broker are often used interchangeably in the literature; brokers are also often called facilitators. Examples of broker agents and broker-based systems include XML-based GMD Broker, OntoBroker [56], SHADE [39], and COINS.

6.2 Matchmaker

A matchmaker agent just pairs requester agents with provider agents by means of matching given requests of requester agents with appropriate advertised services of registered provider agents. In contrast to the functionality of both the broker and mediator it simply returns a ranked list of relevant provider agents to the requesting agent. As a consequence the requester agent has to contact and negotiate with the relevant provider agents themselves for getting the services it desires. This direct interaction between requester agent and selected provider agents is performed independently from the matchmaker. It avoids, for example, data transmission bottlenecks or single point of failure at the matchmaker but increases direct communication overhead between matched requester and provider agents.

Matchmaking gives a requester agent the full choice of selecting a (set of) provider agents a-posteriori out of the result of service matching. In settings with brokers instead, any requester agent's choice can be viewed as a-priori determined by specifying respective constraints in its request for service to the broker. Examples of matchmaker-based Multi-Agent systems and platforms include IMPACT [3; 31], InfoSleuth [58], and RETSINA/LARKS [77; 78] each of which is described in section 7.

When a service-providing agent registers itself with a matchmaker together with a description of its capabilities specified in an agreed ACDL, it is stored as an advertisement and added to the matchmaker's database. Thus, when an agent inputs a request for services, the matchmaker searches its database of advertisements for a service-providing agent that can fill such a request. Requests are filled when the provider's advertisement is sufficiently similar to the description of the requested service (service matching). The matchmaker returns to the requester the list of advertisements and the contact information of the agents whose advertisement matches the request. The requester chooses the most suitable agent for its needs and interacts directly with it to get the requested service. Figure 5 gives an overview of the interaction pattern of this two-part matchmaking process.

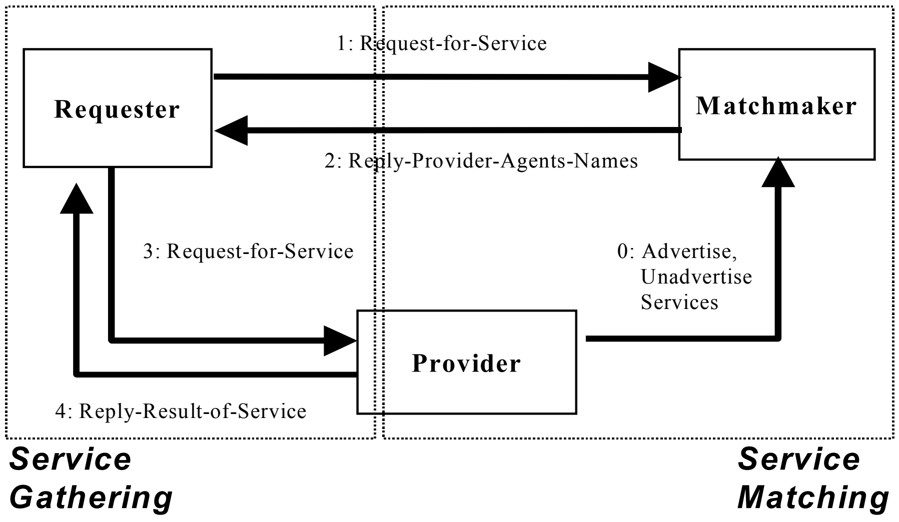


Fig. 5. Interaction pattern of capability and service matchmaking between agents

7. Examples of Capability-Based Coordination

In the following we describe prominent examples of implemented Multi-Agent systems and platforms which are coordinated by middle-agents via the mediation protocols for service matchmaking and brokering, i.e. capability-based coordination. These examples are InfoSleuth, IMPACT, and RETSINA/LARKS. Other related work is surveyed in section 7.4.

It is noteworthy that mediation across Multi-Agent systems boundaries has still received little attention though it is a crucial issue in open environments like the Internet. First steps in this direction include the RETSINA InterOperator [23] between heterogeneous agent societies and Multi-Agent systems.

7.1 InfoSleuth

InfoSleuth [7; 50; 51] has been developed by MCC Inc. in Austin, Texas, USA. It is an agent-based system that can be configured to perform different information management activities in distributed applications such as in an environmental data exchange network and competitive intelligence system. InfoSleuth offers a set of various agents with different roles as follows.

User agents act on behalf of users and interface them to the rest of the agent system, *resource agents* wrap and activate databases and other information sources; *"broker agents"* perform syntactic and semantic matchmaking; *ontology agents* collectively maintain a knowledge base of the different ontologies used for specifying requests and return ontological information on demand; *multi-resource agents* process complex queries that span multiple heterogeneous resources, specified in terms of some domain ontology. Further agent roles concern task planning and execution, monitoring of data streams and system operations, and other specialized functions.

Any provider agent in a given InfoSleuth system announces itself to one or more agents by advertising to it, using the terms and attributes described in a common shared ontology (called "infosleuth ontology") attributes and constraints on the values of those attributes. This special ontology is shared by all agents to use for specifying advertisements and requests. It contains concepts useful to define (a) what kind of *content* is accessible by an agent, (b) what *services* an agent can do for whom, (c) what are the *interfaces* to those services, (d) how well can an agent *perform* those services at the moment, and (e) other properties of an agent such as location, used communication protocols, etc.

Agent capabilities are represented in InfoSleuth at following four levels: (1) the agent *conversations* that are used to communicate about the service, (2) the *interface* to the service, (3) the *information* a service operates over, and (4) the *semantics* of what the service does.

Requester agents formulate request for services in terms of the Infosleuth ontology. The middle agent then matches the request to provider agents whose advertisements correspond to the constraints specified in the request, and returns a recommendation containing those provider agents to the requester agent. Resource agents, whose databases do not maintain data in terms of the common ontology, access special mapping agents to perform appropriate ontology mapping actions.

Given a query over a domain ontology, a query agent coordinates the processing of it by (a) collaborating with one or multiple matchmakers to identify relevant resource agents, (b) decomposing the query into a collection of subqueries each addressed to these agents, and (c) fusing the subquery results into an integrated answer to the original global query. Issues of subquery translation and execution are encapsulated with the respective resource agent.

Unfortunately, the exact and complete process of query decomposition for general queries (not just data base queries) or the process of matching in the InfoSleuth system has not yet been revealed in the literature. However, what is known from the literature is that service matching occurs at different levels: Syntactic, semantic, and pragmatic matching. In InfoSleuth, syntactic matching refers to the process of matching requests on the basis of the syntax of incoming messages which wrap the query; semantic matching refers to the process of matching the internal data structures of and constraints used in request and capability descriptions; pragmatic matching includes considerations such as the performance of the machine the agent is running on and security requirements.

7.2 IMPACT

The IMPACT (Interactive Maryland Platform for Agents Collaborating Together) platform [31] has been developed at the University of Maryland, College Park, USA. It supports Multi-Agent interactions and agent interoperability in an application independent manner. For this purpose IMPACT provides following set of connected servers: (1) Yellow-pages server performs basic matchmaking among requesters and provider agents based on two weighted hierarchies it maintains – a verb and a noun hierarchy of synonyms – and retrieval algorithms to compute similarities between given service specifications, (2) registration server for agent registration and maintaining an agent service index used by the yellow-pages server, (3) type server maintains a hierarchical ontology of standard data types and concepts, as well as (4) a thesaurus and (5) a human interface allowing a human to access all the above servers in an IMPACT system. The IMPACT software provides the infrastructure upon which different IMPACT agents may interact; multiple copies of it may be replicated across the network and get synchronized when needed.

Any service provided by an IMPACT agent is specified in a special markup service description language. A service specification consists of (a) a service name in terms of a verb-noun(noun) expression such as rent:car(Japanese), (b) typed input and output variables, and (c) attributes of services such as usage cost, average response time to requests for that particular service, etc. Services may be either mandatory or discretionary.

7.3 RETSINA/LARKS

The RETSINA (Reusable Task Structure-based Intelligent Network Agents) Multi-Agent infrastructure has been developed at the Carnegie Mellon University in Pittsburgh, USA. It consists of a system of four different reusable agent types that can be adapted to address a variety of different domain-specific problems. *Interface* *agents* interact with the user, receive user input and display results, *task agents* help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents, and *information/resource agents* provide intelligent access to a heterogeneous collection of information sources. *Middle agents* help requesters find providers.

A collection of RETSINA agents forms an open society of reusable agents that self-organize and cooperate in response to task requirements. The RETSINA framework has been implemented in Java and is being used to develop distributed collections of intelligent software agents that cooperate asynchronously to perform goal-directed information retrieval information integration and planning tasks in support of a variety of decision making tasks.

Mediation in the RETSINA Multi-Agent systems basically relies on service matchmaking. The ACDL developed for matchmaking in RETSINA is called LARKS (Language for Advertisement and Request for Knowledge Sharing).

A Matchmaker supports two types of protocols: ``single shot'' and ``monitor'. When an agent sends a single shot request to a Matchmaker, it gets back the list of providers whose advertisements match the request. When an agent sends a monitor query, besides getting the list of matching agents, it also receives notification as soon as one of the providers exits the system, or new providers enter. Agents use one or the other protocol depending on whether they need just a snapshot of the agent landscape or they need to be kept up-to-date on the changes in the system.

Application domain knowledge in agent advertisements and requests can be currently specified as local ontologies written in a specific concept language ITL or by using WordNet. In the LARKS approach advertisements and requests are expressed using the same language. It is assumed that every LARKS specification is wrapped up in an appropriate KQML-like message by the sending agent, indicating whether the message content is to be treated as a request or an advertisement.

A service specification in LARKS is a frame comprised of the following slots: (1) *Context*: Keywords denoting the domain of the description, (2) *Types*: User-defined data types found in the signature definitions, (3) *Input* and *Output*: Input and output parameter declarations defining the signature of the operation, (4) *InConstraints* and *OutConstraints*: Logical constraints on input/output variables (pre-/post conditions), (5) *ConcDescriptions*: Descriptions of used potentially disambiguating words in terms of the concept language ITL, or keyword phrase, and (6) *TextDescription*: A free text description of the agent's capabilities.

Constraints on the input and output of a capability description or request are specified as conjunction of finite Horn clauses. LARKS is fairly expressive and capable of supporting automated inferences for semantically valid partial matches. The implemented matchmaking process for LARKS specifications employs different techniques from information retrieval, AI, and software engineering to compute the syntactical and semantic similarity among advertised and requested agent capability descriptions. The matching engine contains five different filters for (1) keyword-based context matching, (2) TF-IDF based profile comparison, (3) concept-based similarity matching, (4) type-inference rule-based signature matching, and (5) theta-subsumption based constraint matching of finite Horn clauses. Any user may individually configure these filters to achieve the desired tradeoff between performance and matching quality. For more information on LARKS, see [77; 78; 79].

7.4 Other Approaches to Service Description and Matching

Other approaches on (annotated) specification and matching of descriptions of capabilities and services via standardized markup languages, ontologies, and agent capability description languages include SHOE [43], OntoBroker [56], XML Broker, eCo system [24], and JAT/CDL [9], respectively.

7.4.1 JAT/CDL

The capability description language (CDL) has been developed at the university of Edinburgh. Capabilities of and requests for services are described in CDL either in terms of achievable objectives or as performable actions. A request of service is specified in CDL as a task description; the task description in the example would require the capability of the hospital agent described above. For the purpose of automated logic-based reasoning on capability descriptions the statements are formally specified using first-order predicate logic.

Logic-based reasoning over descriptions in CDL is based on the notion of capability subsumption or through instantiation. Capability subsumption in CDL refers to the question whether a capability description can be used to solve a problem described by a given task description in CDL. A capability C subsumes a task T if (1) as a result of performing C all output constraints of T are satisfied, and (2) if in the situation that precedes the result of performing C, all input constraints of C are satisfied. This definition of capability subsumption resembles that of semantic plug-in matching of LARKS descriptions.

Both the CDL and the proposed matching of CDL descriptions has been implemented in Java and tested in selected application scenarios for brokering in Multi-Agent systems developed in JAT (Java Agent Template).

7.4.2 eCo CBL

The eCo system's [24] common business library (CBL) consists of business descriptions and forms represented in an extensible, public set of XML building blocks that service providers in the domain of electronic commerce can customize and assemble. Semantics of the CBL are derived through analysis of industry standards, such as EDI X12850 for specification of purchase orders. This led to a base set of common terms, term mappings in the domain of e-commerce, and corresponding XML data elements, attributes, and generic document type definitions (DTD). Any request for a service is transformed to an appropriate standard DTD, which then can either be used to find similar DTDs of available service descriptions or to produce a stream of information events routed to and processed by business applications.

7.4.3 Techniques and Formalisms Related to Capability Descriptions

The problem of capability and service descriptions can be tackled at least from the following different approaches:

1. Software specification techniques

Agents are computer programs that have some specific characteristics. There are numerous works for software specifications in formal methods, like model-oriented VDM and Z, or algebraic-oriented Larch. Although these languages are good at describing computer programs in a precise way, the specification usually contains too much detail to be of interests to other agents. Besides, those existing languages are so complex that the semantic comparison between the specifications is impossible. The reading and writing of these specifications also require substantial training.

2. Action representation formalisms

Agent capability can be seen as the actions that the agents perform. There are a number of action representation formalisms in AI planning like the classical one STRIPS. The action representation formalisms are inadequate in our task in that they are propositional and not involving data types.

3. Concept languages for knowledge representation

There are various terminological knowledge representation languages. However, ontology itself does not describe capabilities. On the other hand, ontological descriptions provide auxiliary concepts to assist the specification of the capabilities of agents.

4. Database query capability description

The database query capability description technique is developed as an attempt to describe the information sources on the Internet, such that an automated integration of information is possible. In this approach the information source is modeled as a database with restricted querying capabilities.

8. Conclusions

In this chapter we focused on two primary topics. First, we provide a model of what constitutes a MAS infrastructure as a set of services and conventions that allow agents to interoperate. We also present RETSINA as a fully implemented MAS infrastructure that adheres to the proposed model. Second, we focused on Agent and Service Discovery and interoperation. This is crucial for leveraging the power of the Web and allowing agent to automatically interact to perform services.

MASs are more than just a set of agents gathered in the same system, and more than an extension of single agents in some distributed fashion. To work together, agents need a way to find each other, a common communication language, and a shared ontology to understand each other's messages. The role of the MAS infrastructure is to provide location services, ontologies, and language that allow agents to collaborate, exchange information and services. The result is that MASs emerge by the aggregation of agents around an infrastructure which is the "glue" that keeps the agents together, rather than being a by product of the collaboration between agents.

The Internet is an open system where heterogeneous agents can appear and disappear dynamically. As the number of agents on the Internet increases, there is a need to define middle-agents to help agents locate others that provide requested data and services. We introduced several different classes of basic services any middle-agent needs to appropriately instantiate to be able to perform a meaningful, effective, and reliable mediation between agents. We have presented a taxonomy of middle agents along several dimensions and surveyed some prominent examples for Multi-Agent systems and platforms which are coordinated by special kinds of middle-agents, namely broker and matchmaker agents.

9. Acknowledgments

This work is supported by ONR contract N00014-96-1222, and DARPA contracts F30602-98-2-0138 and F30602-00-2-0592. I would like to acknowledge the members of the Intelligent Agents Group at CMU for the development of the RETSINA MAS infrastructure and helpful discussions. The first part of this paper is joint work with Massimo Paolucci, Brent Langley, Martin Van Velsen and Joe Giampapa of the Intelligent Agents Group at CMU. The second part is joint work with Matthias Klusch at DFKI. An extended version of the second part is appearing at M. Klusch and K. Sycara (2001), Brokering and Matchmaking for Coordination of Agent Societies: A Survey In: A. Omicini et al. (eds.), Coordination of Internet Agents, Springer.

References

1. S. Arai, K. Sycara, and T.R. Payne (2000). "Multi-Agent Reinforcement Learning for Scheduling Multiple Goals," In: *ICMAS2000*.
2. Y. Arens, C. A. Knoblock, C. Hsu (1996)."Query Processing in the SIMS Information Mediator," In: Austin Tate (Ed.), *Advanced Planning Technology*, AAAI Press.
3. K. Arisha, T. Eiter, S. Kraus, F. Ozcan, R. Ross, and V.S. Subrahmanian. (2000). "IMPACT: Interactive Maryland platform for agents collaborating together," In: *IEEE Intelligent Systems*, 14(2), 2000.
4. K. S. Barber, D. N. Lam, C. E. Martin, and R. M. McKay (2000). "Sensible Agent Testbed Infrastructure for Experimentation," In: *Agents 2000: Workshop on Infrastructure for scalable MAS*. Barcelona, Spain.
5. A. Barua, A.B. Whinston, and F. Yin (2000). "Value and productivity in the Internet economy," In: *IEEE Computer*, May, 2000.
6. R.J. Brachman, J.G. Schmolze (1985). "An Overview of the KL-ONE Knowledge Representation System," In: *Cognitive Science*, Vol. 9(2): 171—216
7. A. Cassandra, D. Cassandra, M. Nodine (2000). "Capability-based matchmaking," In: *Proceedings Agents-2000 Conference on Autonomous Agents*, Barcelona, ACM Press.
8. C. Castelfranchi, C. (1998). "Modeling social action for AI agents," In: *Applied Artificial Intelligence* 103, 157, 182.
9. CDL. Capability Description Language. http://www.aiai.ed.ac.uk/oplan/cdl/
10. Coabs (2000). Grid Web Site. http://coabs.globalinfotek.com/.
11. Coabs (2000). Corba Web Site. http://corba.org/.
12. K. Decker, K. Sycara, M. Williamson (1997). "Middle-agents for the Internet," In: *Proceedings IJCAI-97 Conference on Artificial Intelligence*, Nagoya.
13. C. Fellbaum (1998) *WordNet: An Electronic Lexical Database*. MIT Press.
14. T. Finin, Y. Labrou, and J. Mayfield (1997). "KQML as an agent communication language," In: J. Bradshaw (ed): *Software Agents*. MIT Press.
15. FIPA Agent Communication Language. http://www.fipa.org/spec/fipa99spec.htm
16. FIPA (2000). Foundation For Physical Agents. http://www.fipa.org/.
17. L. N. Foner (1996). "A Security Architecture for Multi-Agent Matchmaking," In: *ICMAS-96*.
18. S. Funfrockcen (1998). "Transparent Migration of Java-based mobile agents: Capturing and reestablishing state of Java programs," In: *MA98*. Berlin, Germany.

19. H. Garcia-Molina, et al. (1995). "The TSIMMIS Approach to Mediation: Data Models and Languages," In: *Proceedings of Workshop NGITS-95*, ftp://db.stanford.edu/pub/garcia/1995/tsimmis-models-languages.ps
20. L. Gasser (2000). "MAS Infrastructure Definitions, Needs, and Prospects," In: *Agents 2000: Workshop on Infrastructure for scalable MAS*. Barcelona, Spain.
21. M.R. Genesereth, A.M. Keller, O. Duschka (1997). "Infomaster: An Information Integration System, In: *Proceedings of ACM SIGMOD Conference*, May 1997.
22. J.A. Giampapa, O.H. Juarez-Espinoza, K. Sycara (2001). "Configuration Management for Multi-Agent Systems," In: *Proceedings of the Conference on Autonomous Agents 2001*. Montreal CA, June 2001.
23. J.A. Giampapa, M. Paolucci, K. Sycara (2000). "Agent interoperation across Multi-Agent system boundaries," In: *Proceedings Agents-2000 Conference on Autonomous Agents*, Barcelona, ACM Press
24. R.J. Glushko, J.M. Tenenbaum, B. Meltzer (1999). "An XML framework for agent-based e-commerce, In: *Communications of the ACM*, 42(3), March 1999.
25. J.R. Graham and K.S. Decker (2000). "Towards a Distributed, Environment Centered Agent Framework," In: N. Jennings and Y. Lesperance (eds.), *Intelligent Agents IV: Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.
26. M. Greaves, H. Holback, and J. Bradshaw (1999a). "What is a Conversation Policy," In: *Agents 99: Workshop on Specifying and Implementing Conversation Policies*.
27. M. S. Greenberg, J. C. Byington, and D. G. Harper (1998). "Mobile Agents and Security," In: *IEEE Communications*.
28. S. Haustein, S. Luedecke (2000). "Towards information agent interoperability," In: M. Klusch, L. Kerschberg (eds.), *Proceedings of CIA-2000 Workshop on Cooperative Information Agents*, LNAI 1860, Springer
29. A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, Y. Ravid (2000). "Access control meets public key infrastructure, or: Assigning roles to strangers," In: *IEEE Symposium on Security and Privacy*, Oakland, May 2000.
30. G. Huck, P. Fankhauser, K. Aberer, E.J. Neuhold (1998). "Jedi: Extracting and Synthesizing Information from the Web," In: *Proceedings of Conference on Cooperative Information Systems CoopIS'98*, IEEE Computer Society Press.
31. IMPACT. Interactive Maryland Platform for Agents Collaborating Together. http://www.cs.umd.edu/projects/impact/
32. JADE (2000). Programmer's Guide, June 5th, 2000. http://sharon.cselt.it/projects/jade/.
33. M. Jarke, M.A. Jeusfeld, C. Quix, T. Sellis, P. Vassiliadis (2000). "Metadata and data warehouse quality," In: M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis (2000). *Fundamentals of data warehouses*, Springer.
34. JAT Java Agent Template. http://cdr.stanford.edu/ABE/JavaAgent.html
35. S. Jha, P. Chalasani, O. Shehory and K. Sycara (1998). "A Formal Treatment of Distributed Matchmaking," In: *Proceedings of 2nd Conference on Autonomous Agents* (Agents 98), Minneapolis, MN, May, 1998.
36. S. Jini (2000) Jini Web Site. http://www.sun.com/jini.
37. KIF. Knowledge Interchange Format: http://logic.stanford.edu/kif/
38. W. Kim, et al. (1993). "On resolving schematic heterogeneity in multidatabase systems," In: *International Journal on Distributed and Parallel Databases*, Vol. 1:251-279
39. D. Kuokka, L. Harrada (1995). "On using KQML for matchmaking, In: *Proceedings CIKM-95, 3rd Conf. on Information and Knowledge Management*. AAAI/MIT Press.
40. Y. Labrou, T. Finin, Y. Peng (1999). "Agent communication languages: The current landscape," In: *IEEE Intelligent Systems*, March/April.
41. B. Langley, M. Paolucci, and K. Sycara (2001). "Discovery of Infrastructure in Multi-Agent Systems," In: *The Agents 2001 Workshop on Infrastructure for Agents, MAS, and Scalable MAS*.

42. J. S. Liu and K. Sycara (1996). "Multi-Agent Coordination in Tightly Coupled Task Scheduling," In: *ICMAS-96*.
43. S. Luke, L. Spector, D. Rager, and J. Hendler (1997). "Ontology-based web agents," In *1st International Conference on Autonomous Agents*.
44. N. Lynch (1996). *Distributed algorithms*. Morgan Kaufmann.
45. D.W. Manchala (2000) "E-commerce trust metrics and models," In: *IEEE Internet Computing*, 4(2), March/April 2000.
46. D. Martin, A. Cheyer, and D. Moran (1999). "The Open Architecture: A Framework for Building Distributed Software Systems," In*: Applied Artificial Intelligence* 13 (1-2), 92, 128.
47. Y. Mass, O. Shehory (2000). "Distributed trust in open Multi-Agent systems," In: *Proceedings Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies*, June, 2000.
48. P. Mitra S. Decker and S. Melnik. "Framework for the semantic web: An rdf tutorial," In: *IEEE Internet Computing*, November/December 2000.
49. S. Muggleton, L. De Raedt (1994). "Inductive logic programming: Theory and methods," In*: Journal of Logic Programming*, vol 9(20).
50. M. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, A. Unruh (2000). "Active information gathering in InfoSleuth," In: *International Journal of Cooperative Information Systems*, vol. 9(1&2).
51. M. Nodine, W. and A. Ngu (1999). "Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth™," In: *Proceedings of the 15th International Conference on Data Engineering*.
52. H. Nwana, D. Ndumu, L. Lec and J. Collis (1999). "ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems," In: *Applied Artificial Intelligence Journal* 13 (1, 129, 186).
53. OIL. Ontology Interchange Language. http://www.ontoknowledge.org/oil/.
54. OKBC. Open Knowledge Base Connectivity. http://www.ai.sri.com/okbc/.
55. D.E. O'Leary (2000). "Different firms, different ontologies, and no one best ontology," In: *IEEE Intelligent Systems*, September/October.
56. OntoBroker project. http://ontobroker.aifb.uni-karlsruhe.de/.
57. M. Papazoglou, G. Schlageter (1998). *Cooperative information systems*. Academic Press, London.
58. B. Perry, M. Taylor and A. Unruh (1999). "Information Aggregation and Agent Interaction Patterns in Infosleuth," In *cia99*. ACM Press.
59. Rabarijaona et al. (2000). "Building and searching an XML-based corporate memory," In: *IEEE Intelligent Systems*, May/June 2000.
60. RDF (S): XML-based Resource Description Framework Schema Specification. http://www.w3.org/TR/WD-rdf-schema/
61. RETSINA. http://www.cs.cmu.edu/softagents/retsina.html.
62. O. Shehory, and K. Sycara (2000). "The RETSINA Communicator," In: *Proceedings of the Conference on Autonomous Agents 2000*, Barcelona, Spain, July, 2000.
63. O. Shehory, K. Sycara, P. Chalasani, and S. Jha (1998). "Increasing Resource Utilization and Task Performance by Agent Cloning," In: M. S. V. A. Rao and M. Woolridge (eds*.) Lecture Notes in AI: Intelligent Agents*. Springer-Verlag.
64. O. Shehory, K. Sycara, G. Sukthankar, and V. Mukherjec (1999). "Agent Aided Aircraft Maintenance," In: *Agents-99*.
65. A. Sheth, E. Mena, A. Illaramendi, V. Kashyap (1996) "OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies," In: *Proceedings Conf. on Cooperative Information Systems CoopIS-96*. IEEE Computer Society Press.

66. A. Sheth, V. Kashyap, T. Lima (1999). "Semantic information brokering: How can a Multi-Agent approach help?" In: M. Klusch, O. Shehory, G. Weiss (eds.) *Proceedings CIA-1999 Workshop on Cooperative Information Agents*, LNAI 1652, Springer.
67. A. Sheth and J.A. Larson. "Federated database systems," In: *ACM Computing Surveys,* 22(3), 1990.
68. M. P. Singh (1998). "Agent Communication Languages: Rethinking the Principles," In*: IEEE-Computer* 11.
69. I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback (1998). "Designing Conversation Policies Using Joint Intention Theory," In: *ICMAS-98*. IEEE Press.
70. R. G. Smith (1980). "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," In: *IEEE Transactions on computers* 29 (12), 1104 1113).
71. SPKI Simple PublicKey Infrastructure. ftp://ftp.ietf.org/internet-drafts/draft-ietf-spki-cert-theory-02.txt
72. N. Suri, J. M. Bradshaw, P. T. G. Maggie R. Breedy, G. A. Hill, T. S. M. Renia Jerffers, B. R. Pouliot, and D. S. Smith (2000). "NOMADS: toward a strong and safe mobile agent system," In: *Agents 2000*. ACM Press.
73. K. Sycara (1990). "Negotiation Planning: An AI Approach," In: *European Journal of Operational Research* 46, 216-234.
74. K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng (1996). "Distributed Intelligent Agents," In: *IEEE Expert, Intelligent Systems and their Applications* 11 (6), 36 45.
75. K. Sycara, K. Decker, and D. Zeng (1998). "Intelligent Agents in Portfolio Management," In: *Agent Technology: Foundation, Application and Markets*, N. Jennings and M. Wooridge (eds.). Chapter 14, Springer, Fall, 1998.
76. K. Sycara, J. Lu, and M. Klusch (1998). "Interoperability Among Heterogeneous Software Agents on the Internet," Technical Report CMU-RI-TR-98-22, School of Computer Science, Carnegie Mellon University.
77. K. Sycara, J. Lu, M. Klusch, S. Widoff (1999). "Dynamic service matchmaking among agents in open information environments. "ACM SIGMOD Record, Vol. 28, No. 1, March 1999.
78. K. Sycara, M. Klusch, S. Widoff, J. Lu (2001). "LARKS: Dynamic matchmaking among heterogeneous software agents in cyberspace," In: *Journal on Autonomous Agents and Multi-Agent Systems*, March 2001.
79. K. Sycara and D. Zeng (1994). "Towards an Intelligent Electronic Secretary," In*: CIKM-94*.
80. K. Sycara, and D. Zeng (1996). "Coordination of Multiple Intelligent Software Agents," In*: International Journal of Intelligent Cooperative Information Systems*, Vol. 5, Nos. 2 and 3, 1996.
81. M. Tambe (1997). "Towards Flexible Teamwork," In: *Journal of Artificial Intelligence Research* 7, 83 124.
82. J. D. Thomas, K. Sycara, and T. R. Payne (1998). "Heterogeneity, Stability and Efficiency in Distributed Systems," In*: ICMAS1998*.
83. M. Tsvetovat, K. Sycara, Y. Chen, and J. Ying (2000). "Customer Coalitions in the Electronic Marketplace," In: *Proceedings of Workshop on Agent-Mediated Electronic Commerce, Fourth International Conference on Autonomous Agents*.
84. G. Wiederhold (1992). "Mediators in the architecture of future information systems," In: *IEEE Computer Systems*, 25(3), March 1992.
85. H.C. Wong, K. Sycara (1999). "Adding security and trust to Multi-Agent systems," In: *Proceedings Autonomous Agents 1999 Workshop on Deception, Fraud, and Trust in Agent Societies*. May 1999.
86. H.C. Wong, K. Sycara (2000). "A taxonomy of middle-agents for the Internet," In: *Proceedings of ICMAS*-2000, Boston, Mas. July 2000.

87. J. Yamamoto, and K. Sycara (2001). "A Stable and Efficient Buyer Coalition Formation Scheme for E-Marketplaces," In: *Proceedings of the Conference on Autonomous Agents* (Agents 2001), Montreal, Canada, June, 2001.
88. B. Yu, M.P. Singh (2000). "A social mechanism of reputation management in electronic communities," In: *Proceedings CIA-2000 Workshop on Cooperative Information Agents*, M. Klusch, L. Kerschberg (eds.), LNAI 1860, Springer.
89. G. Zacharia, A. Moukas, and P. Macs (1999). "Collaborative Reputation Mechanisms in Online Marketplaces," In: *HICSS-32*.
90. D. Zeng and K. Sycara (1998). "Bayesian Learning in Negotiation," In: *International Journal of Human Computer Systems*, Vol. 48, 1998.

Logical Foundations of Agent-Based Computing

Wiebe van der Hoek[1,2]

[1] Institute of Information and Computing Sciences,
Department of Philosophy
Utrecht University, PO box 80.089 3508 TB Utrecht
The Netherlands
`wiebe@cs.uu.nl`
`http://www.cs.uu.nl/~wiebe`
[2] Department of Computer Science
The University of Liverpool
United Kingdom

Abstract. Logics for agents are useful when specifying, implementing and verifying agent programs. We show that modal logic provides a nice tool to define informational, motivational and dynamic aspects of agents. We conclude by showing how an agent programming language can also benefit from this modal approach.

1 Introduction

There is an increasing interest in applying logical tools to formarlize and analyze agent systems. These tools play a role in at least three levels of software engineering: at the level of *specification*, of *programming* and the level of *verification*. One of the benefits of a logical approach to agents, is that it can unify these three approaches in some way.

Specification As computer-programs and architectures for intelligent systems become more and more complex, in order to understand the behaviour of such systems, designers and users frequently make use of *folk*- or *common psychology*. Using terms like 'belief', 'wish' and 'anger', we are equipped to predict and explain behaviour of ourselves and others. The philosopher Dennett used the phrase *intentional system* to refer to an entity that is best understood in terms of folk-psychology notions such as beliefs, desires, and the like [4]. This was also what Hofstadter was referring to already in '81, when one of his characters, Sandy, puts the following forward in a Coffee House Conversation ([11]):

> But eventually, when you put enough feelingless calculations together in a huge coordinated organization, you'll get something that has properties on another level. You can see it – in fact you *have* to see it– not as a bunch of little calculations, but as a system of tendencies and desires and beliefs and so on. When things get complicated enough, you're forced to change your level of description. To some extend that' already happening, which is why we use words such as "want," "think," "try," and "hope," to describe chess programs and other attempts at mechanical thought.

The intentional stance has been widely discussed in the literature –let us just remark here that Sandy of the Coffeeshop Conversation claims that the really interesting things in AI will only begin to happen, 'when the program *itself* adopts the intentional stance towards itself'– and it is not our intention to add to this debate; see [16] for a discussion and references. What is important here, is that the intentional stance is an abstraction tool, allowing us to specify and reason about complex systems, without having to go into their internal structure. The logics we are going to present in Section 2 are designed with this purpose.

Programming By definition ('agre' means 'actor'), agents are the producers of action. Agents perform actions in order to shape and modify the environment they inhabit. This in fact is already true for mere (intelligent) systems. In 'classical AI' for example, systems are designed according to the so-called *Sense-Plan-Act*-cycle, in which the system observes the state of the environment, then does some deliberation or planning in which a decision about the next action is taken, which is then exercised, after which the effects of it can be observed again, etc. Now, one way to conceive agents is that they enrich the notion of *state* in this cycle; apart from the state of the environment, or the real world, in the agent approach one also assumes a *mental state*.

The most appealing components of such an internal state define the agent's *informational* (like knowledge and belief) and *motivational* (desires, wishes, intentions) attitudes, but one could also think about *normative* or *social* attitudes like obligations and norms or *emotional* attitudes (anger, pain, fear).

It is assumed that agents typically have some kind of awareness or introspective capabilities, they can inspect (parts of) their own mental state. Thus, an agent may know that it does not know some fact, or believe that is has the permission to do some action, or know that it wishes to be without pain. In other words, agents not only (as in the SPA-cycle) observe the environment, but also inspect their own mental state.

Moreover, also the domain of action is no longer restricted to the real world. Agents are pretty well capable of *changing their mind* for instance: they revise their beliefs, modify their goals and can ask for permission. Authors like Shoham ([15]) put it even more bluntly: (the semantics of) an agent action, or program, is nothing else than a transition from one state to another. That some physical agents, like robots, also happen to affect the real world, is in such a conception not more than a 'side effect' of the mental state, or its dynamics.

Verification Having specified a solution for a problem, and implemented a system that should do the job, one needs to show that the implemented specification is correct. Or, more modestly, one is sometimes satisfied if the implemented system satisfies certain properties. Verification often involves temporal properties. Examples of general properties of programs are *safety* (it will never be the case that some undesirable property becomes true) and *liveness* (eventually, some requirements will be met). There are several ways to approach the problem of verification. We give an illustration of this in Section 4.

The rest of this tutorial paper is organised af follows. In Section 2 we make our case for modal logic. Zooming in on epistemic logic, we discuss a number of technical issues. In Section 3, we give an overview of three main approaches to the logic of agent-systems. In Section 4 we demonstrate a toy Agent Programming Language that has more or less separate constructs that address the agents' informational or motivational attitudes, and for which the logics of Section 3 supply at least tools for a neat *semantics* of such programming languages.

2 Intensional Logic

Intensional logic, and, in particular modal logic, have proven to be popular when modelling agents. To explain this, let us consider one characteristic property of classical logic, and that intensional logic typically wants to avoid.

Observation 1 (Extensionality) Let $[q/p]\varphi$ denote the formula φ, but with (an arbitrary number of occurrences of) the subformula p replaced by q. Then, classical logic encompasses the following property:

$$\models (p \leftrightarrow q) \rightarrow (\varphi \leftrightarrow [q/p]\varphi)$$

In words, extensionality says that, to determine the truth-value of a formula φ, it is only the truth-value of its subformulas that counts: if we replace any occurrence of a subformula p by another formula q *with the same meaning*, then this does not matter for the value of the value as a whole. Since the truth-value, or the meaning of a formula is sometimes also denoted as its extension, we can rephrase Observation 1 loosely as: the extension of a complex formula is determined by the extension of its subformulas, not by their form.

To give an example, let p denote that the summerschool is in Prague, and let q be the statement that it is in July. We then quickly recognize that p and q are both true, and thus are equivalent: $(p \leftrightarrow q)$. Furthermore, let l denote that logic is important (true) and w that the summerschool is one week (false). Then, according to extensionality, we have that $(w \rightarrow q)$ is equivalent to $(w \rightarrow p)$, and $l \vee (q \wedge w)$ is equivalent to $l \vee (p \wedge w)$. Combining complex assertions and then calculating their truth-value, is done by substituting the values (extension) for the subformulas. $\text{Ext}(l \vee (q \wedge w)) == \text{Ext}(l \vee (p \wedge w))$

Having established that classical logic satisfies the property of extensionality, one may wonder whether this is desirable, or whether there are constructs in natural language that do not satisfy this principle. It appears there are many. Let c denote that the summerschool is in Czech Republic. Then 'c, because p' is obviously true, whereas 'c, because of q' makes no sense. Noting that, by extensionality, $p \rightarrow c$ and $q \rightarrow c$ are equivalent, we obtain two conclusions: 'B because of A' cannot be modelled by $A \rightarrow B$, and, even stronger, 'because of' cannot be modelled in propositional logic at all (since 'because of' is not extensional, whereas classical logic is). This observation was in fact one of the motivations to develop modal logic.

But there are more example of constructs that are not extensional. For instance ‘I wish the summerschool to be in Prague’ is not the same as ‘I wish the summerschool to be in July’. Also, knowing p is different from knowing q. Compare ‘last year, q’, with ‘last year, p’. ‘When postponing the school for a month, $\neg q$’ does not necessarily mean ‘when postponing the school for a month, $\neg p$’. Thus, when reasoning with motivational attitudes (wishing), informational attitudes (knowing), temporal properties (last year) or hypothetical events (when postponing), we don’t have extensionality. Ergo: we cannot use classical logic to deal with them.

Modal logic is one attempt to circumvent these problems. In a nutshell, the modal language adds one or more unary operators $\Box$ to the language, where $\Box\varphi$ can be used to model ‘φ is known’, or ‘φ is always the case’, ‘φ is a desire’ or ‘φ is a result of executing program π’. The intuition of such formulas is best explained by looking at the semantics of $\Box$. Given a situation s (for the moment, think of it as a truth-assignment to atoms), $\Box\varphi$ is true if φ is true in all situations t that are relevant for s. For instance, is p denotes ‘sunny in Prague’ and q ‘sunny in Amsterdam’, and s is the situation where I am in Prague, where it is sunny, then for me, two situations are relevant, if compatibility with my knowledge is concerned: t_1 in which $p \wedge q$ is true, and t_2 in which $(p \wedge \neg q)$ is. Since in all my alternatives p is true, I know p ($\Box p$), but we also have $\neg\Box q$ and $\neg\Box\neg q$) (see Figure 1). Note that this perfectly solves our problem of extensionality: in s, we have $(p \leftrightarrow q)$ but also $(\Box p \wedge \neg\Box q)$.

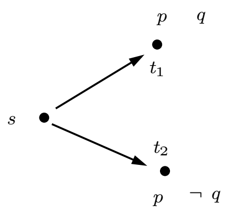


Fig. 1. A Kripke model for knowledge without extensionality

2.1 Modal Logic

All the formalisms in this overview are in fact some kind of modal logic. Mostly, they are *multi-modal*, having several modal operators for different agents, or to capture different mental attitudes. One can have modal operators $K_i\varphi$ for ‘agent i knows that φ, or $B_i\varphi$, for agent i believes that φ (such epistemic operators are the focus of Section 2.2), but in $X\varphi$, the operator X can also be a goal, a wish or a desire for the agent, or even a program (see 3.3).

That is, a Kripke model M will be a tuple $M = \langle S, \pi, \bigcup_{X \in \mathcal{X}} R_X\rangle$ where S is a non-empty set of worlds or states s, π gives, for every state s the truth-value $\pi(s)(p)$ for every atom p, and R_X is a set of accessibility relations, one for each

$X \in \mathcal{X}$. In order to determine whether a formula $\varphi \in \mathcal{L}$ is true in w (if so, we write $(M, w) \models \varphi$), we look at the structure of φ:

$$\begin{array}{ll} M, s \models p & \text{iff } \pi(s)(p) = \mathit{true} \\ M, s \models (\varphi_1 \wedge \varphi_2) & \text{iff } M, s \models \varphi_1 \text{ and } M, s \models \varphi_2 \\ M, s \models \neg\varphi & \text{iff not} M, s \models \varphi \\ M, s \models X\varphi & \text{iff } \forall\, s(R_X st \Rightarrow M, t \models \varphi) \end{array}$$

Under such a definition, we say that X is the necessity operator for an accessibility relation R_X. The clause for $X\varphi$ is sometimes also written in a functional way: $\forall\, t \in R_X(s), M, t \models \varphi$. A formula φ is true in a model, written $M \models \varphi$, if $M, s \models \varphi$ for all $s \in S$. If $\mathcal{M}$ is a class of models, φ is said to be *valid on* $\mathcal{M}$, if for all $M \in \mathcal{M}, M \models \varphi$. This is an interesting notion in modal logic: it appears that many modal properties correspond to some restriction on Kripke models. For instance, the formula $X\varphi \to \varphi$ is valid on the class of models in which R_X is reflexive, i.e., if for all $s \in S, R_X ss$.

If modal logic is used for so many agent attitudes, it is an interesting question what the properties of any modal logic are, i.e., properties φ that are valid in every model. For such φ, we write $\models \varphi$. They are sometimes referred to as instances of the problem of *Logical Omniscience*, since, when interpreted as knowledge, they express that agents are omniscient: they are perfect reasoners.

Definition 2. Let φ, ψ be modal formulae, and let $\mathbf{X}$ be some operator.

- $\models \mathbf{X}\varphi \wedge \mathbf{X}(\varphi \to \psi) \to \mathbf{X}\psi$ $\quad LO1$
- $\models \varphi \Rightarrow \models \mathbf{X}\varphi$ $\quad LO2$
- $\models \varphi \to \psi \Rightarrow \models \mathbf{X}\varphi \to \mathbf{X}\psi$ $\quad LO3$
- $\models \varphi \leftrightarrow \psi \Rightarrow \models \mathbf{X}\varphi \leftrightarrow \mathbf{X}\psi$ $\quad LO4$
- $\models (\mathbf{X}\varphi \wedge \mathbf{X}\psi) \to \mathbf{X}(\varphi \wedge \psi)$ $\quad LO5$
- $\models \mathbf{X}\varphi \to \mathbf{X}(\varphi \vee \psi)$ $\quad LO6$
- $\models \neg(\mathbf{X}\varphi \wedge \mathbf{X}\neg\varphi)$ $\quad LO7$

If X denotes knowledge, $LO1$ for example says that knowledge is closed under consequences. $LO2$ expresses that agents know all validities. If X has to model being a goal, then all the properties (maybe except $LO4$) are rather questionable. We will see later on how in some systems these properties are nevertheless accepted; in others, there are some more complicated constructs for goals. In the next section, we will see examples of modal properties that one can add on top of the properties of Definition 2.

2.2 Epistemic Logic

Epistemic logic is the logic of knowledge, which is of importance to researchers in philosophy, computer science, AI, and game theory. The material in this subsection is completely covered in the text books [5,12]. For the simplest kind of epistemic logic, it is sufficient to enrich the language of classical propositional logic by unary operators K_i, where $K_i\varphi$ stands for "agent i knows φ". Here, an agent may be a human being, a robot, a machine, or simply a 'process'. Before looking at an application, let us note that the meaning (or truth value) of $K_i\varphi$ cannot be given by any propositional truth table in terms of the truth value of φ: knowledge is not extensional. Why are these knowledge operators useful? The derivation and correctness proofs of *communication protocols* form a nice example.

Example 1 (alternating bit protocol). There are two processes, let us say a 'Sender S' and a 'Receiver R'. The goal is for S to read a tape $X = \langle x_0, x_1, \ldots\rangle$, and to send all the inputs it has read to R over a communication channel. R in turn writs his received messages on an output tape Y. Unfortunately the channel is not trustworthy: there is no guarantee that all messages arrive. On the other hand, *some* messages will not get lost: if a message is sent repeatedly, an instance of it will arrive eventually. Now the question is whether one can write a protocol (program) that satisfies the following two constraints:

– *safety:* at any moment, Y is a prefix of X;
– *liveness:* every x_i will eventually be written on Y.

In the protocol below, $K_S(x_i)$ means that Sender knows that the i-th element of X is equal to x_i.

PROTOCOL FOR S:

```
S1 i :=0
S2 while true do
S3     begin read $x_i$;
S4         send $x_i$ until $K_S K_R(x_i)$;
S5         send "$K_S K_R(x_i)$" until $K_S K_R K_S K_R(x_i)$
S6         i := i + 1
S7     end
```

PROTOCOL FOR R:

```
R1 when $K_R(x_0)$ set i :=0
R2 while true do
R3     begin write $x_i$;
R4         send $K_R(x_i)$ until $K_R K_S K_R(x_i)$;
R5         send "$K_R K_S K_R(x_i)$" until $K_R(x_{i+1})$
R6         i := i + 1
R7     end
```

An important aspect of the protocol is that Sender at line $S5$ does not continue reading X and does not yet add 1 to the counter i. We will show why this is crucial for guaranteeing safety. For, suppose that the lines $S5$ and $R5$ would be absent, and that instead line $R4$ would read as $R4'$: `send` $K_R(x_i)$ `until`

$K_R(x_{i+1})$; Suppose also, as an example, that $X = \langle a, a, b, \ldots \rangle$. Sender starts by reading x_0, an a, and sends it to R. We know that an instance of that a will arrive at a certain moment, and so by line $R3$ it will be written on Y. Receiver then acts as it should and sends an acknowledgement ($R4'$) that will also arrive eventually, thus Sender continues with $S6$ followed by $S3$: once again it reads an a and sends it to Receiver. The latter will eventually receive an instance of that a, but will not know how to interpret it: "is this a a repetition of the previous one, because Sender does not know that I know what x_0 is, or is this a the next element of the input tape, x_1"? This would clearly endanger safety.

As a final remark on the protocol, let us note that it is possible to rewrite the protocol without using any knowledge operators. The result is known as the 'alternating bit protocol'.

For negotiations and games it is not only important for participants to know what the others do know, but even more to know what the others do *not* know. Thus, your ignorance can provide me with useful information. A well-known example of this phenomenon is the *wise men* puzzle, in which one wise person can derive the color of his hat from the fact that his colleagues have said they don't know the color of their hats. A somewhat more complex variant of this phenomenon is the muddy children example, discussed below.

It is interesting to investigate notions of group knowledge for multiple agents. For example, for a group of n agents $\{1, \ldots, n\}$, one can define 'Everybody Knows' ($E\varphi$) by $E\varphi \equiv K_1\varphi \wedge \cdots \wedge K_n\varphi$. Another intriguing notion of group knowledge is 'Common Knowledge' ($C\varphi$), that should mean something like $E\varphi \wedge EE\varphi \wedge EEE\varphi \wedge \ldots$. (Unfortunately, such an infinite conjunction is not allowed in the language of epistemic logic.) Common Knowledge is a very strong notion, which therefore holds only of very weak propositions φ. In general it is very difficult to establish Common Knowledge, especially in situations like the following, where communication is not generally known to be totally reliable.

Example 2 (Byzantine generals). Imagine two allied generals, A and B, standing on two mountain summits, with their enemy in the valley between them. It is commonly known that A and B together can easily beat the enemy, but if only one of them attacks, he will certainly lose the battle.

A sends a messenger to B with the message b (= "I propose that we attack on the first day of the next month at 8 PM sharp"). It is not guaranteed, however, that the messenger will arrive. Suppose that the messenger does reach the other summit and delivers the message to B. Then $K_B b$ holds, and even $K_B K_A b$. Will it be a good idea to attack? Certainly not, because A wants to know for certain that B will attack as well, and he does not know that yet. Thus, B sends the messenger back with an 'okay' message. Suppose the messenger survives again. Then $K_A K_B K_A b$ holds. Will the generals attack now? No, because B does not know whether his 'okay' has arrived, so $K_B K_A K_B b$ does not hold, and Common Knowledge of b has not yet been established. One proves that in order to start a coordinated attack, Common Knowledge can never be established in this way using a messenger.

Example 3 (The muddy children). In this example the principal players are a father and k children, of whom m (with $m \leq k$ have mud on their forehead. The father wants to have a serious talk with the muddy children. Thus, he calls all the children together. None of them knows whether it is muddy or not, but they can all accurately perceive the other children and judge whether they are muddy. Moreover, all this is general knowledge; it is also Common Knowledge that all children are perfect logical reasoners and have even successfully finished a course on epistemic logic. Now father has a very simple announcement ψ to make:

> *At least one of you is muddy. If you know that you are muddy, please come forward.*

After this, nothing happens (except in case $m = 1$). When the father notices this, he literally repeats the announcement ψ. Once again, nothing happens (except in case $m = 2$). The announcement and subsequent silence are repeated until the father's m-th announcement. Suddenly all m muddy children step forward! It would go too far to explain the logical techniques needed to give a sound explanation, but one gets a good idea when investigating what happens in the cases $m = 1, 2$. Thus, suppose $m = 1$ and father just announced ψ, then the only muddy child knows it is muddy, because it does not see any muddy companions. It duly steps forward. Now suppose $m = 2$, and call the muddy children m_1 and m_2. Let us follow m_2's reasoning. After the first 'ψ', m_2 reasons about m_1 just like we did in the previous case: "I don't know whether I'm muddy. If not, m_1 wouldn't see any muddy companions and would step forward". At the father's second 'ψ', m_2 knows that m_1 has not in fact stepped forward, so:"m_1 must have seen another muddy child. I don't, so that must have been me". Now m_2 steps forward, and m_1 as well (by a symmetrical argument). Note, finally, that however many children are muddy, there is no Common Knowledge that there is even at least one muddy child before the father makes his first announcement! For example, in case $m = 2$, child m_1 holds it to be possible that it is not muddy and that simultaneously m_2 holds it for possible that m_2 is not muddy either. During the course, we will study the dynamics of the knowledge states of the children involved, and, for a particular case, see how we end up with an appropriate Kripke model that explains why the muddy children step forward.

Epistemics: a formal treatment For convenience, we now give a formal definition of the general logic for knowledge in a group of m agents, starting by giving its language.

Definition 3. Let P be a non-empty set of propositional variables, and $m \in I\!N$ be given. The *language* L is the smallest superset of P such that

$$\varphi, \psi \in \mathsf{L} \Rightarrow \neg\varphi, (\varphi \wedge \psi), K_i\varphi, C\varphi, D\varphi, E\varphi \in \mathsf{L} \ (i \leq m).$$

We also assume to have the usual definitions for $\vee, \leftarrow$ and $\leftrightarrow$ as logical connectives, as well as the special formula $\bot =_{\text{def}} (p \wedge \neg p)$. In the sequel, we will use $\Box$ as a variable over the operators $\mathsf{OP} = \{K_1, \ldots, K_m, \ C, D, E\}$. Indices i and j will range over $\{1, \ldots m\}$.

The intended meaning of $K_i\varphi$ is 'agent i knows φ', $D\varphi$ means 'φ is distributed knowledge', or 'φ is implicit knowledge of the m agents'. $E\varphi$ has to be read as 'everybody knows φ' and $C\varphi$ is 'it is common knowledge that φ'. Distributed knowledge is the knowledge that is implicitly present in a group, and which might become explicit if the agents were able to communicate . For instance, it is possible that no agent knows the assertion ψ, while at the same time $D\psi$ may be derived from $K_1\phi \wedge K_2(\phi \rightarrow \psi)$). A common example of distributed knowledge in a group is for instance the fact whether two members of that group have the same birthday. The meaning of 'everybody knows ϕ' is simply that all members of the group know that ϕ, and Common knowledge of ϕ is supposed to be $E\phi \wedge EE\phi \wedge EEE\phi \wedge \ldots$. The following definition establishes the exact properties of and relations between the notions mentioned.

Definition 4. *The logic $S5_m(CDE)$, or* **L** *for short, has the following axioms:*
$A1$ *any axiomatization for propositional logic*
$A2$ $(K_i\varphi \wedge K_i(\varphi \rightarrow \psi)) \rightarrow K_i\psi$
$A3$ $K_i\varphi \rightarrow \varphi$
$A4$ $K_i\varphi \rightarrow K_iK_i\varphi$
$A5$ $\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$
$A6$ $E\varphi \leftrightarrow (K_1\varphi \wedge \cdots \wedge K_m\varphi)$

$A7$ $C\varphi \rightarrow \varphi$
$A8$ $C\varphi \rightarrow EC\varphi$
$A9$ $(C\varphi \wedge C(\varphi \rightarrow \psi)) \rightarrow C\psi$
$A10$ $C(\varphi \rightarrow E\varphi) \rightarrow (\varphi \rightarrow C\varphi)$

$A11$ $K_i\varphi \rightarrow D\varphi$
$A12$ $(D\varphi \wedge D(\varphi \rightarrow \psi)) \rightarrow D\psi$
$A13$ $D\varphi \rightarrow \varphi$
$A14$ $D\varphi \rightarrow DD\varphi$
$A15$ $\neg D\varphi \rightarrow D\neg D\varphi$

On top of that, we assume the following derivation rules:

$R1 \vdash \varphi, \vdash \varphi \rightarrow \psi \;\Rightarrow\; \vdash \psi$
$R2 \vdash \varphi \;\Rightarrow\; \vdash K_i\varphi$, *for all* $i \leq m$
$R3 \vdash \varphi \;\Rightarrow\; \vdash C\varphi$

In words, we assume a logical system $(A1, R1)$ for rational agents, (that the agents are taken to be rational, is perhaps best reflected by the fact that we have the properties $(\Box\varphi \wedge \Box(\varphi \rightarrow \psi)) \rightarrow \Box\psi$ for all $\Box \in \mathsf{OP}$ —which follows from $A2, A9, A12$ and, in the case of E, with a simple calculation using $A6$). Individual knowledge, common knowledge and distributed knowledge are all supposed to be *veridical* ($A3$, $A7$ and $A13$, respectively). The agents are assumed to be *fully introspective*: they are supposed to have *positive* ($A4$) as well as *negative* ($A5$) introspection; properties we also ascribe to distributed knowledge ($A14$ and $A15$, respectively). Both properties of introspection can be shown to hold for common knowledge as well. Axiom $A6$ can be understood as the definition of E, whereas

$A8$ says that all common knowledge is known by everybody as such. Axiom $A10$ is also known as the *induction axiom*.

What are the appropriate models for this system? Kripke models of the form $M = \langle S, \pi, R_1, \dots, R_m, R_D, R_E, R_C \rangle$ where S and π are as before, and the accessibility relations are given suggestive names indicating to which operators they belong. To achieve the properties of Definition 4 we then require:

– For $i \leq m$, every R_i is an *equivalence* relation. The logic comprising the properties for the individual agents is often referred to as the system **S5**.
– $R_E = \bigcup_{i \leq m} R_i$. φ is known by everybody if no agent considers $\neg\varphi$ possible.
– $R_X = \bigcap_{i \leq m} R_i$. φ is distributed knowledge when true in all the states eliminated by no agent.
– R_C is the transitive reflexive closure of R_E: φ is commonly known if there is no path in the model leading to a $\neg\varphi$ world.

3 Logics for Agency

3.1 Intention Logic

In [3] Cohen and Levesque aim to specify what they call the "rational balance" between the beliefs, goals, plans, intentions, commitments and actions of autonomous agents. The main focus of their paper is on intention, for which they mention the following prerequisite: agents should act on their intentions, not in spite of them; adopt intentions that are believed to be feasible; keep intentions, but not forever; discharge intentions when believed to be satisfied; alter intentions when relevant beliefs change; and adopt intentions during plan formation.

Cohen and Levesque do not want to define intentions in terms of beliefs and desires, but, instead, introduce a third mental state to model them. Following Bratman ([2]), they ascribe to intentions the following functional roles:

(1) Intentions normally pose problems for the agent, who needs to determine ('plan') a way to achieve them;
(2) Intentions provide a "screen of admissibility" for adopting other intentions: one cannot adopt intentions that are incompatible with existing ones;
(3) Agents "track" the success of their attempts to achieve their intentions, giving rise to possibly replanning by the agent.

In their formal language, Cohen and Levesque assume a predicate logic to reason about other agents, a given domain and about time. To this, they add operators: $\mathsf{HAPPENS}\ \alpha$ means that α happens next, $\mathsf{DONE}\ \alpha$ that action α has just happened, $\mathsf{AGT}\ i\ \alpha$ that i is the agent of action α, $\mathsf{BEL}\ i\ \varphi$ that φ is believed and, finally, $\mathsf{GOAL}\ i\ \varphi$ that φ is one of agent i's goals. Actions are either atomic, or a test $\varphi?$ or composed as $\cdot ; \cdot$ (sequential composition), $\cdot \mid \cdot$ (nondeterministic choice or $\cdot^*$ (iteration). Finally, there are also expressions that refer to time, such as $\mathsf{2:30PM}$.

Without giving the full formal semantics, it is helpful to realise that formulas are interpreted on quadruples M, σ, v, n, where M is a Kripke model with accessibility relations B for the beliefs and G for the goals, v gives the interpretation of variables, n is an integer denoting time, and, finally, σ is a sequence of events. To get a feeling for the semantics, we explain the interpretation of several formulas in state $\langle M, \sigma, \mathsf{v}, n\rangle$. First, actions α are interpreted as $[\![\alpha]\!]$ which are sequences: $M, \sigma, \mathsf{v}, n[\![\alpha]\!](n+m)$ means that α is a sequence of events $e_1, e_2, \dots e_m$ 'complying' with σ: $\sigma(n+i) = e_i$. Action constructs are then interpreted as expected: $M, \sigma, \mathsf{v}, n[\![\varphi?]\!]n$ if $M, \sigma, \mathsf{v}, n \models \varphi$. For sequential composition, $M, \sigma, \mathsf{v}, n[\![\alpha;\ \beta]\!]m$ holds iff there is a k such that $M, \sigma, \mathsf{v}, n[\![\alpha]\!]k$ and $M, \sigma, \mathsf{v}, k[\![\beta]\!]m$.

1. $\mathsf{HAPPENS}\ \alpha$ is true if for some $m \geq n$, we have $M, \sigma, \mathsf{v}, n[\![\alpha]\!]m$, i.e, the meaning $[\![\alpha]\!]$ of α describes a sequence that happens after n. There is also a two-placed HAPPENS, $\mathsf{HAPPENS}\ i\ \alpha = \mathsf{HAPPENS}\ a \wedge \mathsf{AGT}\ i\ \alpha$ denoting that i was the actor.
2. $\mathsf{DONE}\ \alpha$ is true if there for some $m \leq n$, $M, \sigma, \mathsf{v}, m[\![\ \alpha\ ]\!]n$. There is also a two-placed done, $\mathsf{DONE}\ i\ \alpha$ denoting that i was the actor.
3. $\mathsf{BEL}\ i\ \varphi$ is true if for all σ' for which $\langle\sigma, n\rangle Bi\sigma'$ one has $M, \sigma', \mathsf{v}, n \models \varphi$.
4. $\mathsf{GOAL}\ i\ \varphi$ is true if for all σ' for which $\langle\sigma, n\rangle Gi\sigma'$ one has $M, \sigma', \mathsf{v}, n \models \varphi$.

Beliefs and goals are ordinary modal operators, and hence satisfy the logical omniscience properties of Definition 2. This, of course, is most questionable for goals. Also, not that goals are *declarative*, not *procedural*: agents have a goal that something is the case, and not to do some action. By additional constraints on the model, [3] guarantee that furthermore, for belief, agents have positive and negative introspection, and always have consistent beliefs: $\models (\mathsf{BEL}\ i\ p) \to \neg(\mathsf{BEL}\ i\ \neg p)$. The connection between the informational and motivational attitude is given by a property called *realism*: $\models (\mathsf{BEL}\ i\ p) \to (\mathsf{GOAL}\ i\ p)$. The motivation Cohen and Levesque give for this property is '...if an agent believes p now, he cannot now want it currently false; agents do not choose what they cannot change. Conversely, if p is now true in all the agent's chosen worlds, then the agent does not believe it is currently false' ([3, page 234]). It seems that this property is better modelled by imposing $\models \mathsf{BEL}\ i\ p \to \neg\mathsf{GOAL}\ i\ \neg p$. Another property in this system is *expected consequences*, denoted by $\models (\mathsf{GOAL}\ i\ p \wedge (\mathsf{BEL}\ i\ (p \to q)) \to (\mathsf{GOAL}\ i\ q)$. Thus, goals are closed under the believed consequences of the agent.

The system leaves room to define temporal modalities. Let *eventually* φ be defined as $\Diamond\varphi \equiv \exists\, x(\mathsf{HAPPENS}\ x;\ \varphi?)$. Note how such a concept can be defined by quantifying over actions: φ is eventually true if it holds after some sequence of events. *Always* φ, $\Box\varphi$ is then defined as the dual of eventually φ: $\Box\varphi = \neg\Diamond\neg\varphi$: always φ holds if there is no sequence of events leading to $\neg\varphi$. Another concept that is defined by quantification is $\mathsf{BEFORE}\ p\ q = \forall\, c(\mathsf{HAPPENS}\ c;\ q? \to (\exists\, a(a \leq c) \wedge \mathsf{HAPPENS}\ a;\ p?$: p is true before q if for every event c that makes q true, there is another event, happening not later than c, that leads to p. As a last temporal modality, $\mathsf{LATER}\ p$ denotes $(\neg p \wedge \Diamond p)$ These temporal operators can be used to express specific persistence properties of agents w.r.t. their goals. For instance, achievement goals are defined as

$$\mathsf{A} - \mathsf{GOAL}\ i\ p = \mathsf{GOAL}\ i\ \mathsf{LATER}\ p \wedge \mathsf{BEL}\ i\ \neg p$$

That is, agent i believes that p is currently false, but in its desired worlds, p is eventually true. Next, agents do not try to achieve the same goal forever: $\models \Diamond\neg(\mathsf{GOAL}\ i\ \mathsf{LATER}\ P)$. Thus, agents eventually drop their achievements goals. The framework is rich enough to capture fanatic agents, having persistent goals:

$$\begin{aligned}\mathsf{P-GOAL}\ i\ p = \quad & \mathsf{GOAL}\ i\ \mathsf{LATER}\ p \wedge \mathsf{BEL}\ i\ \neg p\ \wedge \\ & \mathsf{BEFORE}\ (\mathsf{BEL}\ i\ p \vee \mathsf{BEL}\ i\ \Box\neg p)\ (\neg\mathsf{GOAL}\ i\ \mathsf{LATER}\ p)\end{aligned}$$

In words: p is a persistent goal if it is an achievement goal with the property that, if the agent drops it as a goal, then before that, he came either to believe that the goal is fulfilled, or that p will never become true anymore. Using the second conjunct of the definition of persistent goals, we immediately derive:

$$\mathsf{P-GOAL}\ i\ q \rightarrow \Diamond[\mathsf{BEL}\ i\ q \vee \mathsf{BEL}\ i\ \Box\neg q] \tag{1}$$

That is, persistent goals persist until they are believe to achieved, or believed to have become unachievable. Let us have a look at how intentions are defined. Here, we only look at a procedural definition:

$$\mathsf{INTEND}\ i\ a = \mathsf{P-GOAL}\ i\ [\mathsf{DONE}\ i\ (\mathsf{BEL}\ i\ (\mathsf{HAPPENS}\ a)?);\ a]$$

that is, action a is intended if it is a persistent goal of the agent that he has done a, but not by accident, no, before he does a he is aware of it: he believes, just before doing a, that a will happen.

Have we now met the desiderata of Bratman? Equation (1) guarantees that intentions cause the agent some problems; as long as the agent does not believe that the intended goal is achieved, he believes that it will be brought about. Looking at the specific structure of the action that is intended, the agent will always know which action has to be taken next. For instance, when the action is sequential, say $a;\ b$, then one easily verifies that if i intends to do $a;\ b$, then he intends to do a. Also, he intends to do $\mathsf{DONE}\ a?b$, which means that i just intends to do the second step of his plan just at the right time.

Moving to Bratman's second requirement, that intentions provide the agent with a screen of admissibility, one has:

$$\mathsf{INTEND}\ i\ b \wedge \Box(\mathsf{BEL}\ i\ [\mathsf{DONE}\ \ i\ a \rightarrow \Box\neg(\mathsf{DONE}\ \ i\ b]) \rightarrow \neg\mathsf{INTEND}\ i\ a;\ b \tag{2}$$

In words: if i intends to do b and he believes that doing a is incompatible with doing b, then the agent will not intend to do $a;\ b$. Finally, Bratman demands that agents track the success of their attempts to achieve their intentions. This is expressed by

$$\begin{aligned}&\mathsf{DONE}\ [(\mathsf{INTEND}\ i\ a) \wedge (\mathsf{BEL}\ i\ (\mathsf{HAPPENS}\ a))]?;\ e\ \wedge \\ &(\mathsf{BEL}\ i\ \neg(\mathsf{DONE}\ \ i\ a)) \wedge \neg(\mathsf{BEL}\ i\ \Box\neg(\mathsf{DONE}\ \ i\ a)) \rightarrow \\ &\qquad\qquad\qquad\qquad \mathsf{INTEND}\ i\ a\end{aligned} \tag{3}$$

(3) says: if the agent intends to do a and he believes that he is doing a but some other event e happens, then, if the agent believes he has not done a, but doing a is still possible, then he persists in intending to do a.

3.2 BDI-Agents

In their paper [14], Rao and Georgeff want to give a possible-worlds formalism for agent systems in which attitudes such as Beliefs, Desires and Intentions play a prominent role. A main difference with the approach of Cohen and Levesque form Section 3.1 is that, rather than taking linear sequences of events as the semantic building blocks, the basic semantic entities in [14] are temporal structures, i.e. trees, where the branching within a tree models the alternative choices that the agent has. Rao and Georgeff mention three crucial elements of their formalism:

– Intentions are treated as first-class citizens on a par with beliefs and goals. Thus, one can define different strategies of commitment with respect to the agent's intentions;
– They distinguish between the choice that the agent has over the actions he can perform, and the possibly different outcomes of an action, which are not under control of the agent;
– An interrelationship between beliefs, goals and intentions is specified that avoids many of the problems usually associated with a possible-world formalism.

To understand the formalism of this BDI-approach, one has to distinguish between a local and a global level. Locally, a world in a Kripke model is not a single point without structure anymore, but instead, it is a branching time tree. To be more precise, a *situation* is a point in a time tree, with a branching time future and a single, linear past. One such a situation is depicted in Figure 2. Given a situation, we do not reason about the history, only about the current situation and the future. In the latter, it is important to make the difference between truth on a path, and truth over paths. For instance, one can express that a certain formula φ is true at *some* point in *every* branch. By varying the quantifiers, one thus obtains 4 possible modalities referring to future.

In the syntax, we can make this more precise by distinguishing between local, *state* formulas and *path* formulas. State formulas (evaluated at a state s) are predicates, and, if ϕ, ϕ_1, ϕ_2 are state formulas, so are $\exists\,\phi$, $(\phi_1 \wedge \phi_2)$ and $\neg\phi$ and the modal formulas $\mathsf{BEL}\phi, \mathsf{GOAL}\phi$ and $\mathsf{INTEND}\phi$. Moreover, for every event e, the formulas *succeeds(e), fails(e), done(e)* etc. are also state formulas. Finally, if ψ is a path-formula, then *optional*ψ is a state formula, expressing that there is a path starting from s, in which the path formula ψ is true. A path-formula ψ (evaluated at a path p) is either a state formula ϕ or obtained from path formulas ψ, ψ_1, ψ_2 by applying the Boolean connectives, or the following constructs: $\Diamond\psi$ (in some future state on the path p, ψ holds) $\bigcirc\psi$ (ψ holds in the next state in the branch) and $\psi_1 \mathsf{~U~} \psi_2$ (either ψ_1 remains true for ever on the path, or ψ_2 will become true somewhere and until that point, ψ_1 is true). Furthermore, *inevitable*ψ (in all paths starting in s, ψ holds) is defined as $\neg$*optional*$\neg\psi$ and $\Box\psi$ (in every future point along every path, ψ is defined as $\neg\Diamond\neg\psi$.

On a global level, such situations are accessible to each other on the basis of what the agent believes, desires or intends. Thus, on the global level, in the syntax we have operators $\mathsf{BEL}, \mathsf{GOAL}$ and INTEND which are $\Box$-operators with

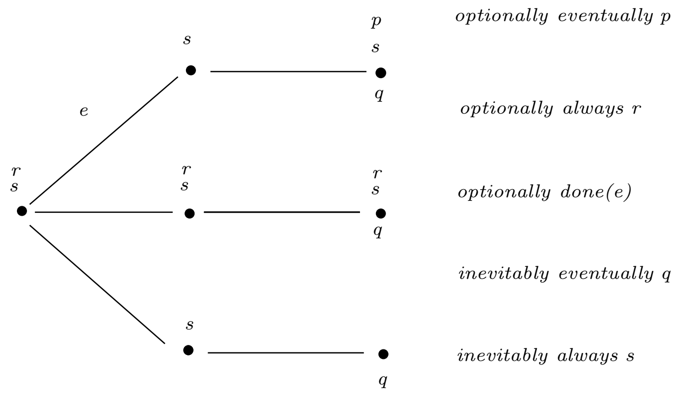


Fig. 2. Path and state formulas, events

respect to relations, or, rather functions $\mathcal{B}, \mathcal{G}$ and $\mathcal{I}$, respectively. To be more precise, let w_t be a situation at time t, with a single past and a branching future. Let X be one of our operators $\mathsf{BEL}, \mathsf{GOAL}$ and INTEND, and, given a situation w_t, $\mathcal{X}_t^w$ its semantic counterpart. Then the truth-definition reads:

$$M, v, w_t \models \mathsf{X}\varphi \text{ iff } \forall\, w' \in \mathcal{X}_t^w, M, v, w'_t \models \varphi$$

Note that agents are aware of the time: they only /believe/desire/intend worlds to be possible with the same timestamp t. Which doe not mean that they can have beliefs, desires and intentions concerning the future: φ can of course refer to a branch or a point later than t. Also note that these beliefs, desires and intentions are defined as modal operators, and thus, suffer from the logical omniscience properties mentioned in 2. By putting constraints on the semantic functions $\mathcal{X}_t^w$, it is guaranteed that beliefs satisfy the KD45 axioms, and for $\mathsf{X} = \mathsf{GOAL}$ or INTEND we have $(\mathsf{X}\varphi \to \psi) \to (\mathsf{X}\varphi \to \mathsf{X}\psi)$ and $(\neg\mathsf{X}\bot)$. As an example of the interrelation between these attitudes, [14] assumes the following scheme for $\langle \mathsf{X}, \mathsf{Y} \rangle \in \{\langle \mathsf{BEL}, \mathsf{INTEND}\rangle, \langle \mathsf{BEL}, \mathsf{GOAL}\rangle, \langle \mathsf{GOAL}, \mathsf{INTEND}\rangle\}$:

$$\mathsf{X}\varphi \to \mathsf{YX}\varphi \qquad (4)$$

Saying that agents are aware of their goals and intentions and, if the agent intends to achieve φ, it has the goal to intend this. Semantically a constraint like (4) is guaranteed by imposing (5)

$$\forall\, w' \in \mathcal{Y}_t^w\, \forall\, w'' \in \mathcal{X}_t^w,\ w'' \in \mathcal{X}_t^{w'} \qquad (5)$$

Concerning goal-intention compatibility, the following constraint (6) is imposed for $\bigcirc$-formulas α, that is, for formulas that have only positive references to optional formulas. It captures the idea that an agent only intends to achieve some of its goals:

$$\mathsf{INTEND}\alpha \to \mathsf{GOAL}\alpha \qquad (6)$$

It is remarkable how Rao and Georgeff divert from Cohen and Levesque. Instead of the realism of the latter, Rao and Georgeff require, for $\bigcirc$-formulas α:

$$\mathsf{GOAL}\alpha \rightarrow \mathsf{BEL}\alpha \tag{7}$$

This property of *belief-goal compatibility* expresses that if an agent has a goal that α –say $\diamond p$– holds, i.e, eventually p, then he believes that this is feasible: the agent thinks that p will indeed eventually be achieved.

Let us end this short overview of BDI agents by showing how several agents w.r.t. their commitments can be defined. First, we define a *blindly* committed agent, an agent that maintains his intentions until he actually believes that he has achieved them. Equation (8) reads as follows: if the agent intends that eventually φ holds, then the agent will maintain this intention until it believes she believes φ (has been achieved):

$$\begin{aligned} &\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi) \rightarrow \\ &\mathit{inevitable}(\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi)\ \mathsf{U}\ \mathsf{BEL}\varphi) \end{aligned} \tag{8}$$

One can relax the blind-commitment strategy to a so-called *single-minded* commitment. In (9), the agent does not hold on to his intentions for ever or until he believes he has achieved them; instead, he maintains his intentions only as long he believe they are still options:

$$\begin{aligned} &\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi) \rightarrow \\ &\mathit{inevitable}(\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi)\ \mathsf{U}\ (\mathsf{BEL}\varphi \vee \neg\mathsf{BEL}\mathit{optional}\varphi)) \end{aligned} \tag{9}$$

Finally, for an *open minded* agent, it is allowed to drop an intention φ already if he drops it as a goal, no matter he believes in φ's implementability:

$$\begin{aligned} &\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi) \rightarrow \\ &\mathit{inevitable}(\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi)\ \mathsf{U}\ (\mathsf{BEL}\varphi \vee \neg\mathsf{GOAL}\mathit{optional}\varphi)) \end{aligned} \tag{10}$$

One can then investigate various properties of the agents that we have just defined. For instance, one can prove that a blindly committed agent satisfies $\mathsf{INTEND}(\mathit{inevitable}\diamond\varphi) \rightarrow \mathit{inevitable}(\diamond\mathsf{BEL}\varphi)$: if such an agent intends that φ will eventually become true, it will inevitably be the case that he eventually will believe that φ holds.

3.3 KARO

We now briefly discuss the KARO framework of van der Hoek, van Linder and Meyer ([10]). This system combines the notion of Knowledge and belief with that of Abilities, Results and Opportunities to do actions. All operators are defined on Kripke models, even Goals, without obtaining the *LO*-properties for them. In order to successfully complete an action, both the opportunity and the ability to perform the action are necessary. Although these notions are interconnected, they are surely not identical: the abilities of agents comprise mental and physical powers, moral capacities, and physical possibility, whereas the opportunity to perform actions is best described by the notion of circumstantial possibility.

The abilities of agents are formalised via the $\mathbf{A}_i$ operator; the formula $\mathbf{A}_i\alpha$ denotes the fact that agent i has the ability to do α. When using the descriptions of opportunities and results as given above, the framework of (propositional) dynamic logic provides an excellent means to formalise these notions. Using events $\mathrm{do}_i(\alpha)$ to refer to the performance of the action α by the agent i, we consider the formulae $\langle \mathrm{do}_i(\alpha)\rangle\varphi$ and $[\mathrm{do}_i(\alpha)]\varphi$. As we shall only encounter deterministic actions in this paper, $\langle \mathrm{do}_i(\alpha)\rangle\varphi$ is the stronger of these formulae; it represents the fact that agent i has the opportunity to do α and that doing α leads to φ. The formula $[\mathrm{do}_i(\alpha)]\varphi$ is noncommittal about the opportunity of the agent to do α but states that if the opportunity to do α is indeed present, doing α results in φ.

Definition 5. Let denumerable sets $\mathrm{A} = \{1, \ldots, n\}$ of agents, Π of propositional symbols and At of atomic actions be given. The language L is the smallest superset of Π such that:

– if $\varphi, \psi \in \mathrm{L}, i \in \mathrm{A}, \alpha \in \mathrm{Ac}$ then $\neg\varphi, \varphi \vee \psi, \mathbf{K}_i\varphi, \langle \mathrm{do}_i(\alpha)\rangle\varphi, \mathbf{A}_i\alpha \in \mathrm{L}$

where Ac is the smallest superset of At such that if $\varphi \in \mathrm{L}$, $\alpha, \alpha_1, \alpha_2 \in \mathrm{Ac}$ then

– $\varphi? \in \mathrm{Ac}$ *tests*
– $\alpha_1;\ \alpha_2 \in \mathrm{Ac}$ *sequential composition*
– $\mathtt{if}\,\varphi\,\mathtt{then}\,\alpha_1\,\mathtt{else}\,\alpha_2\,\mathtt{fi} \in \mathrm{Ac}$ *conditional composition*
– $\mathtt{while}\,\varphi\,\mathtt{do}\,\alpha\,\mathtt{od} \in \mathrm{Ac}$ *repetitive composition*

Actions and knowledge are all defined on the same states. With this, one can for instance formulate properties like

– $K_i[\mathrm{do}_i(\alpha)]\varphi \rightarrow [\mathrm{do}_i(\alpha)]K_i\varphi$ *(perfect recall)*
– $[\mathrm{do}_i(\alpha)]K_i\varphi \rightarrow K_i[\mathrm{do}_i(\alpha)]$ *(no learning)*

In the KARO framework,the mental attitudes of the agents are *dynamic*: apart form actions that change the world (i.e., the truth-value of atomic propositions), they also can affect the state of the agent. We give an example of the effects two mind-changing actions. By expanding his beliefs, the agent just adds a formula to his beliefs, but when revising them, he is careful that his beliefs remain consistent, whenever possible.

Proposition 1. *For all propositional formulas φ, ψ, ϑ we have:*

– $\models [\mathrm{do}_i(\mathtt{revise}\,\varphi)]\mathsf{B}\varphi$
– $\models [\mathrm{do}_i(\mathtt{revise}\,\varphi)]\mathsf{B}\vartheta \rightarrow [\mathrm{do}_i(\mathtt{expand}\,\varphi)]\mathsf{B}\vartheta$
– $\models \neg\mathsf{B}\neg\varphi \rightarrow ([\mathrm{do}_i(\mathtt{expand}\,\varphi)]\mathsf{B}\vartheta \leftrightarrow [\mathrm{do}_i(\mathtt{revise}\,\varphi)]\mathsf{B}\vartheta)$
– $\models \mathbf{K}_i\neg\varphi \leftrightarrow [\mathrm{do}_i(\mathtt{revise}\,\varphi)]\mathsf{B}\bot$
– $\models \mathbf{K}_i(\varphi \leftrightarrow \psi) \rightarrow ([\mathrm{do}_i(\mathtt{revise}\,\varphi)]\mathsf{B}\vartheta \leftrightarrow [\mathrm{do}_i(\mathtt{revise}\,\psi)]\mathsf{B}\vartheta)$

The first clause of Proposition 1 states that agents believe φ as the result of revising their beliefs with φ. Secondly, a revision with φ results in the agent believing at most the formulae that it would believe after expanding its beliefs with φ. Instead of rephrasing all the properties, let us here remark that KARO provides utilities to express the well-known postulates ([1]) for belief revision *within the language*.

To formalise the knowledge of agents on their practical possibilities, [10] introduces the so-called Can-predicate and Cannot-predicate.

Definition 6. *The Can-predicate and the Cannot-predicate are, for all agents i, actions α and formulae φ, defined as follows.*

- $\mathbf{PracPoss}_i(\alpha, \varphi) =^{\text{def}} \langle \text{do}_i(\alpha)\rangle\varphi \wedge \mathbf{A}_i\alpha$
- $\mathbf{Can}_i(\alpha, \varphi) =^{\text{def}} \mathbf{K}_i\mathbf{PracPoss}_i(\alpha, \varphi)$
- $\mathbf{Cannot}_i(\alpha, \varphi) =^{\text{def}} \mathbf{K}_i\neg\mathbf{PracPoss}_i(\alpha, \varphi)$

Thus the Can-predicate and the Cannot-predicate express the agent's knowledge about its practical possibilities and impossibilities, respectively. Therefore these predicates are important for the agent's planning of actions.

In KARO, an agent's goals are not primitive but induced by his wishes. Basically, an agent selects among its (implicit and passive) wishes those that it (explicitly and actively) aims to fulfil. Given the rationality of agents, these selected wishes should be both unfulfilled and implementable. In KARO, wishes ($\mathbf{W}_i\varphi$) are defined as an ordinary modal operator, and the practical possibility $\diamond_i\varphi$ to obtain φ is defined as:

$$\text{M}, s \models \diamond_i\varphi \Leftrightarrow \exists\, k \in \mathbb{N}\, \exists\, a_1, \ldots, a_k \in \text{At}(\text{M}, s \models \mathbf{PracPoss}_i(a_1;\ \ldots;\ a_k, \varphi))$$

A goal $\mathbf{Goal}_i\varphi$ in KARO is defined as a wish that is selected ($\mathbf{C}_i\varphi$), not fulfilled yet but implementable:

$$\mathbf{Goal}_i\varphi =^{\text{def}} \mathbf{W}_i\varphi \wedge \neg\varphi \wedge \diamond_i\varphi \wedge \mathbf{C}_i\varphi$$

This definition of goals guarantees that none of the logical omniscience problems of Definition 2 applies. Also for the motivational attitudes, KARO provides for means to update them. That is, agents can commit and uncommit themselves to execute certain tasks. Technically, in the semantics, this is taken care of by adding the notion of an *agenda* for every agent, in every state. Rather than going into the details, we mention some obtained properties of these dynamics:

Proposition 2. *For all $i \in \text{A}$, $\alpha, \beta \in \text{Ac}$ and $\varphi \in \text{L}$ we have:*

1. $\models \mathbf{Committed}_i\alpha \rightarrow \neg\mathbf{A}_i\texttt{commit\_to}\,\beta$
2. $\models \mathbf{Committed}_i\alpha \leftrightarrow \langle\text{do}_i(\texttt{uncommit}\,\alpha)\rangle\neg\mathbf{Committed}_i\alpha$
3. $\models (\mathbf{C}_i\varphi \leftrightarrow \mathbf{K}_i\mathbf{C}_i\varphi) \rightarrow (\mathbf{A}_i\texttt{uncommit}\,\alpha \leftrightarrow \mathbf{K}_i\mathbf{A}_i\texttt{uncommit}\,\alpha)$
4. $\models \mathbf{Committed}_i\alpha \wedge \neg\mathbf{Can}_i(\alpha, \top) \rightarrow \mathbf{Can}_i(\texttt{uncommit}\,\alpha, \neg\mathbf{Committed}_i\alpha)$

In the first item it is stated that being committed prevents an agent from having the ability to (re)commit. Item 2 states that being committed is a necessary and sufficient condition for having the opportunity to uncommit. In item 4 it is stated that agents are (morally) unable to undo commitments to actions that are still known to be correct and feasible to achieve some goal. In item 5 it is formalised that agents know of their abilities to uncommit to some action. The last item states that whenever an agent is committed to an action that is no longer known to be practically possible, it knows that it can undo this impossible commitment.

The following proposition formalises some of the desiderata for the statics of commitments that are valid in KARO:

Proposition 3. *For all* $i \in \mathrm{A}$, $\alpha, \alpha_1, \alpha_2 \in \mathrm{Ac}$ *and all* $\varphi \in \mathrm{L}$ *we have:*

1. $\models \mathbf{Committed}_i\alpha \to \mathbf{K}_i\mathbf{Committed}_i\alpha$
2. $\models \mathbf{Committed}_i(\alpha_1;\ \alpha_2) \to \mathbf{Committed}_i\alpha_1 \wedge \mathbf{K}_i[\mathrm{do}_i(\alpha_1)]\mathbf{Committed}_i\alpha_2$
3. $\models \mathbf{Committed}_i\mathtt{if}\,\varphi\,\mathtt{then}\,\alpha_1\,\mathtt{else}\,\alpha_2\,\mathtt{fi} \wedge \mathbf{K}_i\varphi \to \mathbf{Committed}_i(\varphi?;\ \alpha_1)$

4 Agent Programming

We have now seen how the concept of agents is made more precise by means of *logical systems*. The exact relation of these logics with more practical approaches remains unclear, however, to this day. Several efforts to bridge the gap have been attempted. In particular, a number of *agent programming languages* have been developed for this purpose [13]. These languages show a clear family resemblance with one of the first agent programming languages Agent-0 [15,8], and also with the language ConGolog [6]. Here, we present a brief description of the language GOAL [7]; with its declarative notion of a goal, it fits well in this overview.

As in most agent programming languages, GOAL agents select actions on the basis of their current mental state. A mental state is a pair $\langle\sigma,\gamma\rangle$ where σ are the beliefs and γ are the goals of the agent. Constraint on mental states insist that an agent cannot have a goal to achieve ϕ if the agent already believes that ϕ is the case. Formally, the constraint on mental states $\langle\sigma,\gamma\rangle$ means that no $\psi \in \gamma$ can be inconsistent nor can ψ be entailed by σ ($\sigma \not\models \psi$), and σ must be consistent.

To express conditions on mental states, the language $\mathcal{L}_M$ of mental state formulas is introduced. The language $\mathcal{L}_M$ consists of boolean combinations of the basic mental state formulas $\mathsf{B}\phi$, which expresses that ϕ is believed to be the case, and $\mathsf{G}\psi$, which expresses that ψ is a goal of the agent.

Besides beliefs and goals, a third basic concept in GOAL is that of an agent *capability*. The capabilities of an agent consist of a set of so called *basic actions* which are interpreted as updates on the agent's belief base. An example of a capability is the action $\mathsf{ins}(\phi)$ which inserts ϕ in the belief base. The capabilities of an agent do not modify the agent's goals. Two special actions $\mathsf{adopt}(\phi)$ and $\mathsf{drop}(\phi)$ are introduced to respectively adopt a new goal or drop an old goal. We use *Bcap* to denote the set of all belief update capabilities of an agent. The set of all capabilities is then defined by: $Cap = Bcap \cup \{\mathsf{adopt}(\phi), \mathsf{drop}(\phi) \mid \phi \in \mathcal{L}_0\}$.

A *GOAL agent* is a triple $\langle \Pi, \sigma_0, \gamma_0 \rangle$ that consists of the specification of an *initial mental state* $\langle \sigma_0, \gamma_0 \rangle$ and a set of actions built from the capabilities associated with the agent. Actions derived from the capabilities are *conditional actions* of the form $\varphi \to do(\mathsf{a})$, where $\mathsf{a} \in Cap$, and $\varphi \in \mathcal{L}_M$ is a mental state condition. The mental state condition specifies when the action a may be considered for execution by the agent. Note the most salient differences between GOAL agents and, for example, AgentSpeak or 3APL agents: whereas AgentSpeak and 3APL agents have planning capabilities (by means of plan rules), GOAL agents do not; whereas GOAL agents have declarative goals, neither AgentSpeak nor 3APL has such goals.

One of the key ideas in the semantics for GOAL is to incorporate into the semantics a particular *commitment strategy* (cf. Section 3). The semantics is based on a particularly simple and transparent commitment strategy, called *blind commitment*. An agent that acts according to a blind commitment drops a goal if and only if it believes that that goal has been achieved. By incorporating this commitment strategy into the semantics of GOAL, a default commitment strategy is built into agents. It is, however, only a default strategy and a programmer can overwrite this default strategy by means of the drop action. It is not possible, however, to adopt a goal ψ in case the agent believes that ψ is already achieved.

Formally, the update on the current mental state - and not just the belief base - due to an action is derived from a given partial transition function $\mathcal{T}$ of type : $Bcap \times \wp(\mathcal{L}) \to \wp(\mathcal{L})$. $\mathcal{T}$ specifies how a capability updates a belief base. The update function $\mathcal{M}$ on mental states $\langle \sigma, \gamma \rangle$ is derived from $\mathcal{T}$. Like $\mathcal{T}$, $\mathcal{M}$ is a *partial* function representing the fact that an action may not be executable or *enabled* in some mental states. The semantic function $\mathcal{M}$ maps an agent capability and a mental state to a new mental state. The capabilities of an agent thus are *mental state transformers*.

$$\boxed{\begin{array}{l}
\mathcal{M}(\mathsf{a}, \langle \sigma, \gamma \rangle) = \langle \mathcal{T}(\mathsf{a}, \sigma), \gamma \setminus \{\psi \in \gamma \mid \mathcal{T}(\mathsf{a}, \sigma) \models \psi\} \rangle \\
\text{ for } \mathsf{a} \in Bcap \text{ if } \mathcal{T}(\mathsf{a}, \sigma) \text{ is defined,} \\
\mathcal{M}(\mathsf{a}, \langle \sigma, \gamma \rangle) \text{ is undefined for } \mathsf{a} \in Bcap \text{ if } \mathcal{T}(\mathsf{a}, \sigma) \text{ is undefined,} \\
\mathcal{M}(\mathsf{drop}(\phi), \langle \sigma, \gamma \rangle) = \langle \sigma, \gamma \setminus \{\psi \in \gamma \mid \psi \models \phi\} \rangle, \\
\mathcal{M}(\mathsf{adopt}(\phi), \langle \sigma, \gamma \rangle) = \langle \sigma, \gamma \cup \{\phi\} \rangle \text{ if } \sigma \not\models \phi \text{ and } \not\models \neg\phi, \\
\mathcal{M}(\mathsf{adopt}(\phi), \langle \sigma, \gamma \rangle) \text{ is undefined if } \sigma \models \phi \text{ or } \models \neg\phi.
\end{array}}$$

The second idea incorporated into the semantics concerns the *selection of conditional actions*. A conditional action $\varphi \to do(\mathsf{a})$ may specify conditions on the beliefs as well as conditions on the goals of an agent. As is usual, conditions on the beliefs are taken as a precondition for action execution: only if the agent's current beliefs entail the belief conditions associated with φ the agent will select a for execution. The goal condition, however, is used in a different way. To make this discussion more precise, we introduce a formal definition of a *formula ϕ that partially fulfils a goal in a mental state* $\langle \sigma, \gamma \rangle$, notation: $\phi \rightsquigarrow_\sigma \gamma$. By definition, we have $\phi \rightsquigarrow_\sigma \gamma$ iff for some $\psi \in \gamma : \psi \models \phi$ and $\sigma \not\models \phi$ Formally, for a mental state $\langle \sigma, \gamma \rangle$, the semantics of a mental state formula is defined by:

$$\boxed{\begin{array}{l} \langle\sigma,\gamma\rangle \models \mathsf{B}\phi \text{ iff } \sigma \models \phi \qquad \langle\sigma,\gamma\rangle \models \mathsf{G}\psi \text{ iff } \psi \rightsquigarrow_\sigma \gamma \\ \langle\sigma,\gamma\rangle \models \neg\varphi \text{ iff } \langle\sigma,\gamma\rangle \not\models \varphi \; \langle\sigma,\gamma\rangle \models \varphi_1 \wedge \varphi_2 \text{ iff } \langle\sigma,\gamma\rangle \models \varphi_1 \text{ and } \langle\sigma,\gamma\rangle \models \varphi_2. \end{array}}$$

Now we know what it means that a mental condition holds in a mental state, we are able to formalise the selection and execution of a conditional action by an agent. In the definition below, we assume that the action component Π of an agent $\langle\Pi,\sigma_0,\gamma_0\rangle$ is fixed. The execution of an action gives rise to a *computation step* formally denoted by the transition relation $\xrightarrow{b}$ where b is the conditional action executed in the computation step. More than one computation step may be possible in a current state and the step relation $\longrightarrow$ thus denotes a *possible* computation step in a state. A computation step updates the current state and yields the next state of the computation. Note that because $\mathcal{M}$ is a partial function, a conditional action can only be successfully executed if both the condition is satisfied and the basic action is enabled.

Definition 7. (action selection) Let $\langle\sigma,\gamma\rangle$ be a mental state and $b = \varphi \rightarrow do(\mathsf{a}) \in \Pi$. Then, as a rule, we have that if the mental condition φ holds in $\langle\sigma,\gamma\rangle$, i.e. $\langle\sigma,\gamma\rangle \models \varphi$, and a is enabled in $\langle\sigma,\gamma\rangle$, i.e. $\mathcal{M}(\mathsf{a},\langle\sigma,\gamma\rangle)$ is defined, then $\langle\sigma,\gamma\rangle \xrightarrow{b} \mathcal{M}(\mathsf{a},\langle\sigma,\gamma\rangle)$ is a possible computation step. The relation $\longrightarrow$ is the smallest relation closed under this rule.
We say that a conditional action b is *enabled* in a mental state $\langle\sigma,\gamma\rangle$ in case $\mathcal{M}(\mathsf{a},\langle\sigma,\gamma\rangle)$ is defined.

Temporal Logic for GOAL The semantics of GOAL agents is derived directly from the operational semantics as presented above. The meaning of a GOAL agent consists of a set of so-called *traces*, infinite computation sequences of consecutive mental states and actions performed in those mental states.

Definition 8. A trace s is an infinite sequence $s_0, b_0, s_1, b_1, s_2, \ldots$ where s_i are states, b_i are conditional actions, and for every i we have: $s_i \xrightarrow{b_i} s_{i+1}$, or b_i is not enabled in s_i and $s_i = s_{i+1}$.

An important assumption in the programming logic for GOAL is a *fairness* assumption. In a fair trace, there always will be a future time point at which an action is scheduled (considered for execution) and so a fair trace implements the weak fairness assumption. However, note that the fact that an action is scheduled does not mean that the action also is enabled (and therefore, the selection of the action may result in an idle step which does not change the state).

By definition, the *meaning of a GOAL agent* $\langle\Pi,\sigma_0,\gamma_0\rangle$ is the set of *fair* traces S such that for $s \in S$ we have $s_0 = \langle\sigma_0,\gamma_0\rangle$.

Basic Action Theories and Hoare Triples The specification of basic actions provides the basis for the programming logic and, as we will show below, is all we need to prove properties of agents. Because they play such an important role in the proof theory of GOAL, the specification of the basic agent capabilities requires special care. In the proof theory of GOAL, Hoare triples of the form

$\{\varphi\}\ b\ \{\psi\}$, where φ and ψ are *mental state formulas*, are used to specify actions. The use of Hoare triples ([9]) in a formal treatment of traditional assignments is well-understood: such triples are used to specify the *preconditions*, the *postconditions* (effects) and the *frame conditions* of actions.

Definition 9. A *Hoare triple for conditional actions* $\{\varphi\}\ b\ \{\psi\}$ means that for all traces $s \in S_A$ and time points i, we have that $(\varphi[s_i] \wedge b = b_i \in s) \Rightarrow \psi[s_{i+1}]$ where $b_i \in s$ means that action b_i is taken in state i of trace s.
A *Hoare triple for basic capabilities* $\{\varphi\}\ \mathsf{a}\ \{\psi\}$ means that for all σ, γ

– if $\langle\sigma,\gamma\rangle \models \varphi$ and a is enabled in $\langle\sigma,\gamma\rangle$, then $\mathcal{M}(\mathsf{a}, \langle\sigma,\gamma\rangle) \models \psi$, and
– if $\langle\sigma,\gamma\rangle \models \varphi$ and a is *not* enabled in $\langle\sigma,\gamma\rangle$, then $\langle\sigma,\gamma\rangle \models \psi$.

We first list some general properties of the belief and goal modalities. First, from the definition of the semantics of the goal modality, it is easy to see that $\mathsf{B}\phi \rightarrow \neg\mathsf{G}\phi$ is valid. In particular, an agent never has the goal to achieve a tautology, that is, $\neg\mathsf{G}(\mathsf{true})$ is valid. By definition of a mental state, an agent also cannot have an inconsistent goal. As a consequence, we have that $\neg\mathsf{G}(\mathsf{false})$ is an axiom.

The goal modality is a weak logical operator. In particular, it does not distribute over implication, and we do not have $(\mathsf{G}p \wedge \mathsf{G}(p \rightarrow q)) \rightarrow \mathsf{G}q$. This is due to the fact that two independent goals $\gamma = \{p, p \rightarrow q\}$ are adopted or it is due to the fact that the agent believes q. In the case that $\mathsf{B}q$, we even do not have $\mathsf{G}(p \wedge (p \rightarrow q)) \rightarrow \mathsf{G}q$ because $\mathsf{B}q \rightarrow \neg\mathsf{G}q$. From a given goal $\mathsf{G}\phi$ it is possible to conclude that the agent also has goal $\mathsf{G}\psi$ if ψ is entailed by ϕ and the agent does not already believe that ψ is the case (otherwise the axiom $\mathsf{B}\phi \rightarrow \neg\mathsf{G}\phi$ would be contradicted). Finally, we *cannot* conclude from two goals $\mathsf{G}\phi$ and $\mathsf{G}\psi$ that an agent has the conjunctive goal $\mathsf{G}(\phi \wedge \psi)$. That is, $(\mathsf{G}\phi \wedge \mathsf{G}\psi) \rightarrow \mathsf{G}(\phi \wedge \psi)$ is *not* valid. In sum, most of the usual problems that many logical operators for motivational attitudes suffer from do not apply to our G operator (cf. also Sections 2 and 3).

Conditional actions and capabilities are formalised by means of Hoare triples in our framework. A set of rules to derive Hoare triples for capabilities from *Cap* and conditional actions is listed below.

Rule for Infeasible Capabilities:
$$\frac{\varphi \rightarrow \neg \mathit{enabled}(\mathsf{a})}{\{\varphi\}\ \mathsf{a}\ \{\varphi\}}$$

Rule for Conditional Actions:
$$\frac{\{\varphi \wedge \psi\}\ \mathsf{a}\ \{\varphi'\}, (\varphi \wedge \neg\psi) \rightarrow \varphi'}{\{\varphi\}\ \psi \rightarrow \mathit{do}(\mathsf{a})\ \{\varphi'\}}$$

Consequence Rule:
$$\frac{\varphi' \rightarrow \varphi, \{\varphi\}\ \mathsf{a}\ \{\psi\}, \psi \rightarrow \psi'}{\{\varphi'\}\ \mathsf{a}\ \{\psi'\}}$$

Conjunction Rule:
$$\frac{\{\varphi_1\}\ b\ \{\psi_1\}, \{\varphi_2\}\ b\ \{\psi_2\}}{\{\varphi_1 \wedge \varphi_2\}\ b\ \{\psi_1 \wedge \psi_2\}}$$

Disjunction Rule:
$$\frac{\{\varphi_1\}\ b\ \{\psi\}, \{\varphi_2\}\ b\ \{\psi\}}{\{\varphi_1 \vee \varphi_2\}\ b\ \{\psi\}}$$

The default commitment strategy can also be captured by a Hoare triple. In case $\mathsf{a} \neq \mathsf{drop}(\psi)$, we have that $\{\mathsf{G}\phi\}\ \varphi \to do(\mathsf{a})\ \{\mathsf{B}\phi \vee \mathsf{G}\phi\}$ which expresses that after execution of an action an agent either believes it has achieved ϕ or it still has the goal ϕ in case ϕ was a goal before action execution. The next Hoare triple formalises a similar kind of statement for the absence of goals. In principle, no other action than an adopt action can add a goal to the goals of an agent. However, in case an agent believes that ϕ has been achieved before an action is executed, but after execution no longer believes this to be the case, it may adopt ϕ as a goal again. Formally, we have $\{\neg\mathsf{G}\phi\}\ b\ \{\neg\mathsf{B}\phi \vee \neg\mathsf{G}\phi\}$. Adopting goals again when it is believed they are not established anymore, provides for a mechanism similar to that of maintenance goals.

The remaining axioms and derivation rules concern the special actions drop and adopt. Neither of these actions changes anything with respect to the current beliefs of an agent. This is captured by the following four Hoare triples:

– $\{\mathsf{B}\phi\}\ \mathsf{adopt}(\psi)\ \{\mathsf{B}\phi\}$, $\{\neg\mathsf{B}\phi\}\ \mathsf{adopt}(\psi)\ \{\neg\mathsf{B}\phi\}$,
– $\{\mathsf{B}\phi\}\ \mathsf{drop}(\psi)\ \{\mathsf{B}\phi\}$, $\{\neg\mathsf{B}\phi\}\ \mathsf{drop}(\psi)\ \{\neg\mathsf{B}\phi\}$.

Concerning the changes to goals, if an agent does not believe ψ and ψ is not a contradiction, then $\mathsf{adopt}(\psi)$ results in a (new) goal $\mathsf{G}\psi$. Formally, $\{\neg\mathsf{B}\psi\}\ \mathsf{adopt}(\psi)\ \{\mathsf{G}\psi\}$. An adopt action does not have any effect on current goals of the agent. That is, if ϕ is a goal before the execution of an adopt action, it is still a goal after the execution of the adopt action: $\{\mathsf{G}\phi\}\ \mathsf{adopt}(\psi)\ \{\mathsf{G}\phi\}$. On the other hand, an $\mathsf{adopt}(\psi)$ action does not result in the adoption of a new goal ϕ in case ϕ is not entailed by ψ (similar rules can be formulated for the drop-action):

$$\frac{\not\models \psi \to \phi}{\{\neg\mathsf{G}\phi\}\ \mathsf{adopt}(\psi)\ \{\neg\mathsf{G}\phi\}}$$

Temporal logic On top of the Hoare triples for specifying basic actions, a temporal logic is used to specify and verify properties of a GOAL agent. The temporal logic language $\mathcal{L}_T$ based on $\mathcal{L}$ is defined by: (i) **init** $\in \mathcal{L}_T$, (ii) if $\phi \in \mathcal{L}$, then $\mathsf{B}\phi, \mathsf{G}\phi \in \mathcal{L}_T$, (iii) if $\varphi, \psi \in \mathcal{L}_T$, then $\neg\varphi, \varphi \wedge \psi \in \mathcal{L}_T$, (iv) if $\varphi, \psi \in \mathcal{L}_T$, then φ **until** $\psi \in \mathcal{L}_T$.

init is a proposition which states that the agent is at the beginning of execution, i.e. nothing has happened yet. The **until** operator is a weak until operator. φ **until** ψ means that ψ eventually becomes true and φ is true until ψ becomes true, or ψ never becomes true and φ remains true forever. The usual abbreviations for the propositional operators $\vee$, $\to$, and $\leftrightarrow$ are used. In case we just write false as a formula, this should be taken as an abbreviation for $\mathsf{B}(p \wedge \neg p)$ for some p. The *always* operator $\Box\varphi$ is an abbreviation for φ **until** false, and the *eventuality* operator $\Diamond\varphi$ is defined as $\neg\Box\neg\varphi$ as usual. As already was explained in the previous section, the atoms $\mathsf{B}\phi$, $\mathsf{G}\psi$ and any other state formula are evaluated with respect to mental states. The semantics of temporal formulas (φ **until** ψ is defined as in Section 3.2), relative to a trace s and time point i is defined by:

$$\boxed{\begin{array}{l} s,i \models \textbf{init} \text{ iff } i=0 \quad s,i \models \mathsf{B}\phi \text{ iff } \mathsf{B}\phi[s_i] \;\; s,i \models \mathsf{G}\phi \text{ iff } \mathsf{G}\phi[s_i] \\ s,i \models \neg\varphi \text{ iff } s,i \not\models \varphi,\; s,i \models \varphi \wedge \psi \text{ iff } (s,i \models \varphi \text{ and } s,i \models \psi) \end{array}}$$

For a set of traces S, we define: (i) $S \models \varphi$ iff $\forall\, s \in S, i(s,i \models \varphi)$, and (ii) $\models \varphi$ iff $S \models \varphi$ where S is the set of all traces. Temporal formulas evaluated with respect to the traces of a GOAL agent express properties of that agent. Let A be a GOAL agent and S_A be the set of traces associated with A. Then, if $S_A \models \varphi$, φ is a property of A.

In general, two important types of temporal properties are distinguished. Temporal properties are divided into *liveness* and *safety* properties. Liveness properties concern the progress that an agent makes and express that a (good) state eventually will be reached. Safety properties, on the other hand, express that some (bad) states are never entered. In the rest of this section, we discuss a number of specific liveness and safety properties of an agent $A = \langle \Pi_A, \sigma_0, \gamma_0 \rangle$ and show how these properties can be proven on the basis of the program text only. The fact that proofs can be constructed from just the program text is important because it avoids the need to reason about individual traces of a program. Reasoning from the program text is more economical since the number of traces associated with a program in general is exponential in the size of the program.

The first property we discuss concerns a safety property. Informally, the property states that if φ ever becomes true, then it remains true until ψ becomes true. Formally, this property can be written as $\varphi \rightarrow (\varphi \textbf{ until } \psi)$, which is abbreviated as: $\varphi \textbf{ unless } \psi = \varphi \rightarrow (\varphi \textbf{ until } \psi)$. **unless** properties of an agent A can be proven by proving Hoare triples for conditional actions in Π_A only. In case we can prove that after execution of an arbitrary action either φ persists or ψ becomes true, we can conclude that φ **unless** ψ.

Theorem 1. $\forall\, b \in \Pi_A(\{\varphi \wedge \neg\psi\}\ b\ \{\varphi \vee \psi\})$ *iff* $S_A \models \varphi \textbf{ unless } \psi$

An important special case of an **unless** property is φ **unless** false, which expresses that if φ ever becomes true, it will remain true. The Hoare triples which are needed to prove φ **unless** false simplify to $\{\varphi\}\ b\ \{\varphi\}$. In case we also have **init** $\rightarrow \varphi$, where **init** denotes the initial starting point of execution, φ is always true and φ is an *invariant* of the program.

Liveness properties involve eventualities stating that some state will be reached given some condition. To express a special class of such properties, we introduce the operator φ **ensures** ψ which informally means that condition φ guarantees the realisation of ψ. The operator **ensures** is defined by: $\varphi \textbf{ ensures } \psi = \varphi \textbf{ unless } \psi \wedge (\varphi \rightarrow \Diamond\psi)$. ¿From this operator, below we derive a somewhat less constrained operator 'leads to'. Again, we can show that **ensures** properties can be derived by inspecting the program text only.

Theorem 2. $\forall\, b \in \Pi_A(\{\varphi \wedge \neg\psi\}\ b\ \{\varphi \vee \psi\}) \wedge \exists\, b \in \Pi_A(\{\varphi \wedge \neg\psi\}\ b\ \{\psi\}) \Rightarrow S_A \models \varphi \textbf{ ensures } \psi$

5 Conclusion

We have argued that Modal logic is a convenient tool to reason about, specify, implement and verify intelligent agents. As an illustrative case, we discussed epistemic modal logic. We have described three logical approaches to agents, all dealing with informational and motivational attitudes of agents.

We then presented a simple agent programming language that uses many of the constructs and ideas introduces earlier. Admittedly, there still is a gap between the logical approaches on the one hand, and the agent programming languages on the other. However, first steps are made to bridge this gap.

References

1. C.E. Alchourrón, P. Gärdenfors and D. Makinson, On the logic of theory change: partial meet contraction and revision functions, in *Journal of Symbolic Logic*, **50**, 1985 pp. 510–530.
2. M. Bratman, Intentions, Plans and Practical Reason. Harvard University Press, Cambridge, MA, 1987.
3. P. Cohen and H. Levesque, Intention is Choice with Commitment, in *Artificial Intelligence*, **42** pp. 213–261 (1990).
4. D.C. Dennet, *The Intentional Stance*, MIT Press, 1987.
5. R. Fagin, J.Y. Halpern, Y. Moses and M.Y. Vardi, *Reasoning About Knowledge*, MIT Press, 1995.
6. G. De Giacomo, Y Lespérance and H. Levesque, ConGolog, a Concurrent Programming Language Based on the Situation Calculus, in *Artificial Intelligence*, accepted for publication.
7. K.V. Hindriks, F.S. de Boer, W. van der Hoek and J.-J.Ch. Meyer, *Agent Programming with Declarative Goals* To appear in the proceedings of ATAL'2000.
8. K. Hindriks, F. de Boer, W. van der Hoek and J.-J.Ch. Meyer, Agent Programming in 3APL, in *Autonomous Agents and Multi-Agent Systems*, **2**:4, pp. 357-401 1999.
9. C.A.R. Hoare, *Communicating Sequential Processes* Prentice Hall, 1985.
10. W. van der Hoek, B. van Linder and J.-J. Ch. Meyer, 'An integrated Modal Approach to Rational Agents', in M. Wooldridge and A. Rao (eds.) *Foundations of Rational Agency* Kluwer, Dordrecht, 1999, pp. 37 - 75.
11. D.R. Hofstadter, "Metamagical Themas: A coffeehouse conversation on the Turing test to determine if a machine can think", in *Scientific American*, (1981), pp. 15–36.
12. J.-J.Ch. Meyer and W. van der Hoek, *Epistemic Logic for AI and Computer Science*, Cambridge University Press, 1995.
13. A.S. Rao, AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language, in W. van der Velde and J.W. Perram (eds), *Agents Breaking Away*, 1996.
14. A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall (eds) *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 473–484. Morgan Kaufmann, 1991.
15. Y. Shoham, Agent-oriented programming, in *Artificial Intelligence*, **60**, pp. 51-92, 1993.
16. M. Wooldridge, *Reasoning about Rational Agents*, MITP, 2000.

Standardizing Agent Communication

Yannis Labrou

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, Maryland 21250
jklabrou@csee.umbc.edu

Abstract. An Agent Communication Language (ACL) is a collection of speech-act-like message types, with agreed-upon semantics, which facilitate the knowledge and information exchange between software agents. From Knowledge Query and Manipulation Language (KQML) to FIPA ACL, ACL's have been a cornerstone for the development of systems of communicating agents, and simultaneously they have been the subject of intensive standardization efforts.
Standardization's goal is usability. As a result, although the initial focus on ACL's revolved around establishing the semantics of ACL's, a variety of usability-related questions have entered the picture of standardizing communication among agents. In this article, we present these questions and the work that addresses them, alongside the historical evolution of ACL's, their semantics and the results of their standardization.

1 Introduction

In the past ten years we have experienced a transformation of the term agents from a Artificial Intelligence term to a buzzword, often accompanied by promises of "amazing" feats that agents will be capable of. The term agents by which we always mean software agents now refers to a paradigm for software development, rather than a class of artifacts, which emphasizes autonomy both at design time and runtime, adaptivity, and cooperation; the agent paradigm seems appealing in a world of networked, distributed and heterogeneous systems. This transformation has inevitably affected the tools and methodologies used for software agent development, even though none of the promises has yet been delivered anywhere near fully.

A persistent theme throughout agents' conceptual evolution has been their ability to interact (communicate) with one another and thus be able to tackle collectively problems that no single agent can, individually. For a large part of the agents' community, the role of endowing agents with the ability to communicate was left to the Agent Communication Language (ACL). Knowledge Query and Manipulation Language (KQML), conceived in the early 90's gradually defined the concept of an ACL. KQML sprang out of the work of the Knowledge Sharing Effort (KSE), a consortium led by (mostly) AI researchers, which was aimed at achieving interoperability between knowledge bases. Initially, agents were not part of the vocabulary of the KSE and KQML was thought of as an integral part of a larger solution to the interoperability question (see Section 2); KQML, as its name probably suggests, had nothing to do with agents. Early KQML can be summarized as a collection of speech-act-like message types, expressed as ASCII strings, with a LISP-like syntax, that are transported over TCP/IP connections, and aimed at knowledge and information exchange between software systems that are viewed as Virtual Knowledge Bases. KQML specifications and discussions about KQML at that time, manifested the AI-influenced thinking behind its design and the considerations that were deemed important. It is only 3-4 years after KQML's inception that it was recasted [18,23] as a language for agent communication.

As a larger number of researchers (and users) became part of the fledging KQML community, KQML was dissociated from the waning KSE (and its concerns) and new issues came at the forefront. Many different groups designed and built multi-agent systems that used KQML for inter-agent communication and a large part of their efforts revolved around building the necessary infrastructure for sending and receiving properly formed KQML messages over the network, developing schemes for naming agents and mechanisms for distributing and sharing agent names and (perhaps most importantly) doing all that with traditional (non-AI) programming environments. Initially in C, and soon thereafter in Java, which was just emerging in the mid-90's, these implementations acted as experiments for resolving all these "little" things that the KQML specification had not addressed. As more time was spent dealing with these (seemingly "lower-level") issues, researchers realized that the list of issues was an ever expanding one. During the same period, the term "agents" came into usage to refer to a software-design paradigm rather than AI artifacts. As a result, integrating (and "implementing") KQML with primarily object-oriented code became another major issue.

These changes resulted to a shift of focus from what content (knowledge) the agent (Virtual Knowledge Base, in the KSE vocabulary) represents and which attitudes about that content are to be conveyed (old view, which inevitably emphasizes proper semantics of the communication primitives) to the act of communication itself and considerations about transport, concurrency, networking and architectural assumptions. In the older, earlier framework, the emphasis was on the semantics of the ACL's message types because the semantics properly define the attitudes suggested by the message types; an ACL is exactly that, i.e., a language of attitudes about content. This trend is manifested by the majority of the earlier work on ACLs [9,23]. The later framework, though, emphasizes software development and focuses on how these attitudes are to be communicated among pieces of software (agents) that populate the network; the new focus is syntax and encoding, pragmatic considerations and software integration.

In discussing the standardization of Agent Communication Languages, we first trace the historic origins of KQML; we believe that situating KQML in the context of the Knowledge Sharing Effort will help the reader to better understand the evolution of ACL ideas and concepts. We briefly present KQML and introduce the concepts necessary for discussing agent communication languages [1]. The semantics of KQML have been the single most important issue in the early debates over ACLs, and we include a brief overview of the arguments. The second ACL we discuss is FIPA ACL. This is the language developed by the Foundation for Intelligent Physical Agents, the first organized effort focusing on developing standards in the broader area of agents. In our comparative evaluation of KQML and FIPA ACL, we argue that the developer is indifferent to the subtle semantic-layer differences. The ACL concept, as an agent development tool has evolved in a way that transcends the semantics and the concept now includes new issues. After presenting the new problems challenging ACL research, we examine the current work that addresses them.

Our thesis is that standardization of Agent Communication Languages, has, in fact, come to refer to standardizing Agent Communication, because other topics, some of which we explore here, are an integral part of designing and building challenging systems of communicating agents.

2 Origins of Agent Communication Language Concepts

Understanding the evolution of the Agent Communication Language concept requires understanding the context that gave birth to KQML. KQML was first introduced as one of the results of the Knowledge Sharing Effort (KSE [2]) [31] [34], which has influenced current efforts in inter-agent communication approaches.

The KSE was initiated as a research effort circa 1990 with encouragement and relatively modest funding from U.S. government agencies (DARPA especially). The KSE was highly active for roughly five years thereafter, and enjoyed the participation of dozens of researchers from both academia and industry; the researchers represented various branches of the AI community. Its goal was to develop techniques, methodologies and software tools for knowledge sharing and knowledge reuse between knowledge-based (software) systems, at design, implementation, or execution time. Agents, especially intelligent agents, are an important kind of such knowledge-based systems (other kinds include expert systems or databases, for example). The central concept of the KSE was that knowledge sharing requires communication, which in turn, requires a common language; the KSE focused on defining that common language

In the KSE model, agents (or, more generally, knowledge-based systems) are viewed as (virtual) knowledge bases that exchange propositions using a language that expresses various propositional attitudes. Propositional attitudes are three-part relationships between

- an agent,
- a content-bearing proposition (for example, *it is raining*), and

[1] We use the abbreviation ACL to refer both to an agent communication language as a concept and to ACLs collectively. There is an ACL simply named ACL, but we hope that context will prevent confusion.

[2] http://www.cs.umbc.edu/kse/

– a finite set of propositional attitudes an agent might have with respect to the proposition (for example, believing, asserting, fearing, wondering, hoping, and so on).

For example, $< a, fear, raining_{now} >$ is a propositional attitude.

The KSE model includes three layers of representation: (1) specifying propositional attitudes; (2) specifying propositions (i.e., *knowledge*) - this is often called the (propositional) content layer; and (3) specifying the ontology [20](i.e., vocabulary) of those propositions. The KSE accordingly includes a component (with associated language) for each of these: Knowledge Query and Manipulation Language (KQML) for propositional attitudes, Knowledge Interchange Format (KIF [3]) [17] for propositions, and Ontolingua [14].

Within the KSE approach, the three representational layers are viewed as mainly independent of another. In particular, the language for propositional content (i.e., the content language) can be chosen independently from the language for propositional attitudes. In other words, in the KSE approach, the role of an ACL, namely KQML's in the case of the KSE (or FIPA ACL's, much later) is only to capture propositional attitudes, regardless of how propositions are expressed, even though propositions are what agents are "talking" about [4].

KQML was influenced by the Virtual knowledge Base concept which emphasized propositional content and focused on defining the propositional attitudes. KQML was not concerned with all of the "mechanics" of the interaction/communication nor prescribed much about how an agent is designed and how communication is incorporated in this design. These were intentional choices of the original KQML specification. The intent was to allow for various specific design choices on these issues. Even the chosen syntax was deemed as changeable. But as we will show, building functioning KQML-speaking agents required agreement on certain choices, often on issues seemingly as obscure, as what is, for example, the terminating character of the ASCII stream that is a KQML message transmitted over a TCP connection.

3 KQML: Concepts of ACL's

Existing ACLs are KQML [1] [24], its many dialects and variants, and FIPA ACL. KQML illustrates the basic concepts of all these. With the exception of ACL, a KQML variant that assumes KIF as the content language, all KQML dialects and FIPA ACL follow the basic concepts of KQML that we discuss here.

KQML is a high-level, message-oriented communication language and protocol for information exchange independent of content syntax and applicable ontology. Thus, KQML is independent of the transport mechanism (TCP/IP, SMTP, IIOP, or another), independent of the content language (KIF, SQL,

[3] http://logic.stanford.edu/kif/ and http://www.cs.umbc.edu/kif/

[4] In a similar spirit, the approach of the technical committee that worked on FIPA ACL is that the content language should be viewed as orthogonal to the rest of the ACL message type.

STEP, Prolog, or another), and independent of the ontology assumed by the content.

Conceptually, we can identify three layers in a KQML message: content, communication, and message:

– The content layer bears the actual content of the message in the program's own representation language. KQML can carry any representation language, including languages expressed as ASCII strings and those expressed using binary notation. Every KQML implementation ignores the content portion of the message, except to determine where it ends.
– The communication layer encodes a set of features to the message that describe the lower-level communication parameters, such as the identity [5] of the sender and recipient, and a unique identifier associated with the communication.
– The message layer, which encodes a message that an application would like to transmit to another, is the core of KQML. This layer determines the kinds of interactions one can have with a KQML-speaking agent. The message layer's function is to identify the speech act [2] or performative that the sender attaches to the content. This speech act indicates whether the message is an assertion, a query, a command, or any other of a set of known performatives[6]. In addition, since the content is opaque to KQML, the message layer also includes optional features that describe the content language, the ontology it assumes, and some type of description of the content, such as a descriptor naming a topic within the ontology. These features make it possible for KQML implementations to analyze, route, and properly deliver messages whose content is inaccessible.

3.1 Syntax and Performatives

The syntax of KQML is based on the familiar s-expression used in Lisp -that is, a balanced parenthesis list. The initial element of the list is the performative; the remaining elements are the performative's arguments as keyword/value pairs. The syntax reveals the roots of the initial implementations, which were done in Common Lisp; it has turned out to be quite flexible.

[5] The *identity* is some symbolic name, e.g., *labrou*. One of the first identified problems was that these symbolic names provide no information about a program's actual network address and they can only be useful in the context of some existing name space, to which there is no reference in the name itself.

[6] KQML designers adopted the term performative to mean any of KQML primitive message types. In speech act theory, a performative is an utterance that succeeds simply because the speaker says or asserts it. In English, such utterances typically appear in a first-person, present-tense, declarative form, often accompanied by *hereby* for example, *I hereby request you to turn on the computer*. Cohen has argued [38] that performative is a poor term to use for all ACL primitive message types, because not all can be construed as actions that the sender can make so just by sending them. For historical reasons, however, we continue to use the term for KQML.

A KQML message from agent "grosof" representing a query about the type of processor of a particular laptop may be encoded as shown in Figure 1a. In this message, the KQML performative is *ask-one*, the content is *(CPU libretto50 ?processor)*, the ontology assumed by the query is identified by the token laptop, the receiver of the message is to be an agent identified as *laptop-center*, and the query is written in a language called *KIF*. The value of the `:content` keyword is the content layer; the values of the `:reply-with`, `:sender`, and `:receiver` keywords form the communication layer; and the performative name with the `:language` and `:ontology` keywords form the message layer. In due time, *laptop-center* might send *labrou* the KQML message in Figure 1b.

```
(ask-one
        :sender   labrou
        :receiver laptop-center
        :content  (CPU libretto50 ?processor)
        :ontology electronics
        :language kif)
(a)

(tell
        :sender   laptop-center
        :receiver labrou
        :content  (CPU libretto50 pentium)
        :ontology electronics
        :language kif)
(b)
```

Fig. 1. Examples of messages in KQML: (a) a query from agent "grosof" about the type of CPU of a laptop and (b) a possible response.

Although KQML has a predefined set of reserved performatives, it is neither a minimal required set nor a closed one. A KQML agent may choose to handle only a few (perhaps one or two) performatives. The original KQML specification considered the set to be extensible; a community of agents might choose to use additional performatives if they agree on their interpretation. However, an implementation that chooses to implement one of the reserved performatives must implement it in the agreed-upon way.

One of the design criteria for KQML was to produce a language that could support a wide variety of interesting agent architectures. Thus, KQML introduces a small number of performatives that agents use to describe capabilities; it also introduces a special class of agents called communication facilitators, which are (ordinary) KQML-speaking agents that are capable of processing the aforementioned set of performatives. A facilitator is an agent that performs various useful communication services, such as maintaining a registry of service names, forwarding messages to named services, routing messages based on content, matchmaking between information providers and clients, and providing mediation and translation services.

3.2 Semantics

During its first few years of use, KQML existed with only an informal semantic description. Critics identified this as one of its shortcomings [9]. During the past few years, researchers have put forth several efforts to provide a formal semantics.

In other works [23] [22] [27], Labrou and Finin provide the semantics of KQML in terms of preconditions, postconditions, and completion conditions for each performative.

Assuming a sender A and a receiver B, preconditions indicate the necessary states for an agent to send a performative, **Pre(A)**, and for the receiver to accept it and successfully process it, **Pre(B)**. If the preconditions do not hold, the most likely response will be one of the performatives *error* or *sorry*.

Postconditions describe the states of the sender after the successful utterance of a performative, and of the receiver after the receipt and processing of a message but before a counter- utterance. Postconditions **Post(A)** and **Post(B)** hold unless a *sorry* or an *error* is sent as a response to report the unsuccessful processing of the message.

A completion condition for the performative, Completion, indicates the final state, after, for example, a conversation has taken place and the intention associated with the performative that started the conversation has been fulfilled.

Establishing the preconditions for a performative does not guarantee its successful execution and performance. The preconditions only indicate what can be assumed to have been the state of the interlocutors involved in an exchange, just before it occurred (assuming conforming interlocutor agents). Similarly, the postconditions describe the states of the interlocutors assuming the successful performance of the communication primitive. Preconditions, postconditions, and completion conditions describe states of agents in a language of mental attitudes (*belief*, *knowledge*, *desire*, and *intention*) and action descriptors (for sending and processing a message). No semantic models for the mental attitudes (Bel, Want, Know, Int) are provided, but the language used to describe agents' states severely restricts the ways the mental attitudes can be combined to compose agents' states.

Figure 2 shows an example of semantics for sender A and receiver B in this framework. This semantics for tell suggests that an agent cannot offer unsolicited information to another agent. We can easily amend this by introducing another performative let's call it *proactive-tell*- that has the same semantic description as tell but with Pre(A) being Bel(A,X), and an empty Pre(B).

Another semantic approach [9] [38] builds on earlier work on defining rational agency [8]. The suggested approach views the language's reserved message types as attempts at communication. These attempts involve two or more rational agents that (temporarily) form teams to engage in co-operative communication. This approach strongly links the ACL semantics to the agent theory assumed for the agents involved in an ACL exchange.

tell(A,B,X) A states to B that A believes the content to be true.

- BEL(A,X)
- **Pre(A)**: BEL(A,X) $\wedge$ KNOW(A,WANT(B,KNOW(B,S)))
  Pre(B): INT(B,KNOW(B,S))
  where S may be any of BEL(B,X), or $\neg$(BEL(B,X)).
- **Post(A)**: KNOW(A,KNOW(B,BEL(A,X)))
  Post(B): KNOW(B,BEL(A,X))
- **Completion**: KNOW(B,BEL(A,X))

Fig. 2. KQML semantics for *tell*

What the KQML Specification Did Not Specify

The first KQML specification introduced basic concepts about the language, some of the design assumptions, the syntax for the performatives, natural language descriptions of the meaning of the performatives and some examples. The community of KQML's users was expected to provide for all the other necessary elements for an implementationally viable language. It took the KQML community years to establish which were all the necessary elements.

The semantics were not part of the original KQML specification. Also, the specification did not offer much in terms of valid sequences of messages during agent interaction. Although the specification sporadically eluded to possible response performatives following a particular performative, such commentary was mostly intuitive suggestions. [23] was a first attempt to address these two issues, including an effort to specify *legal* sequences of performatives during agent interaction, using the term *conversation policies* to refer to such sequences. Conversation Policies (or conversation protocols, or just conversations) eventually became a separate thread of research in the ACL community for reasons we discuss later.

Some other issues were more practical in nature but equally important for agent development. The KQML specification assumed the existence of a name space, thanks to which, agents could be referenced by symbolic names; these symbolic names were presumably all that was needed for messages to reach their destination agents. No details of the name space were provided, nor was it explained how the associations between symbolic names and network addresses became known to new agents. Furthermore, just knowing a network address is not sufficient for initiating a KQML interaction, since an agent needs to know at least the network protocol (TCP, SMTP) that the receiving agent can process. Different groups made different choices on these (and many other) issues, since the KQML specification offered no prescribed way for addressing them.

In summary, the early KQML specification [1] described a concept (the Agent Communication Language) and a framework for future work; by no means, was it a specification for a working prototype or an implementation. Agent designers were implicitly asked to make their choices on a number of issues as they saw fit, as long as they stayed within the realm of the conceptual framework.

The exchange of ideas through research meetings and mailing lists established a shared understanding of the issues, even though particular solutions never made it into an official specification. A revised proposed specification made available in 1997 [24], was an attempt to provide a new starting point for discussions and to reconcile many of the then current ideas with the need for a more clear and consistent specification. At around the same time, the advent of the Foundation for Intelligent Physical Agents (FIPA) brought new life to the standardization question.

4 The Foundation for Intelligent Physical Agents (FIPA)

The Foundation for Intelligent Physical Agents is a nonprofit association whose purpose is to promote the success of emerging agent-based applications and services. FIPA's goal is to make available specifications that maximize interoperability across agent-based systems. As this description suggests, FIPA is a standards organization in the area of software agents. The organization originally included the word *Physical* in its name to include agents of the robotic variety. Over time, however, the adjective's presence has come to serve as a reminder that physical -that is, human- agents and interaction with them are part of the association's scope.

FIPA operates through the open international collaboration of member organizations, which are companies and universities active in the field. European and Far Eastern technology companies have been among the earliest and most active participants, including Alcatel, British Telecom, France Telecom, Deutsche Telecom, Hitatchi, NEC, NHK, NTT, Nortel, Siemens, and Telia.

FIPA's operations center around annual rounds of specification deliverables. The current specification is available at the FIPA home page [7]. FIPA assigns tasks to technical committees, each of which has primary responsibility for producing, maintaining, and updating the specifications applicable to its tasks. The technical committee most important within the scope of this article is the one charged with producing a specification for an ACL. In addition, the agent management committee covers agent services such as facilitation, registration, and agent platforms; the agent/software interaction committee covers integration of agents with legacy software applications. Together, these three committees form the backbone of the FIPA specifications.

4.1 FIPA ACL

FIPA's agent communication language, like KQML, draws on speech act theory: messages are actions or communicative acts, as they are intended to perform some action by virtue of being sent. The FIPA ACL specification consists of a set of message types and the description of their pragmaticsthat is, the effects on the mental attitudes of the sender and receiver agents. The specification

[7] http://www.fipa.org

describes every communicative act with both a narrative form and a formal semantics based on modal logic. It also provides the normative description of a set of high-level interaction protocols, including requesting an action, contract net, and several kinds of auctions.

FIPA ACL is superficially similar to KQML. Its syntax is identical to KQML's except for different names for some reserved primitives. Thus, it maintains the KQML approach of separating the outer language from the inner language. The outer language defines the intended meaning of the message; the inner, or content, language denotes the expression to which the interlocutors' beliefs, desires, and intentions, as described by the meaning of the communication primitive, apply.

In FIPA ACL, the communication primitives are called communicative acts, or CA's for short. Despite the difference in naming, KQML performatives and FIPA ACL communicative acts are the same kind of entity. To avoid confusion, we will use the terms performative, (communication) primitive, and communicative act interchangeably.

The FIPA ACL specification document claims that FIPA ACL (like KQML) does not make any commitment to a particular content language. This claim holds true for most primitives. However, to understand and process some FIPA ACL primitives, receiving agents must have some understanding of Semantic Language, or SL. We will discuss this important point later.

4.2 FIPA ACL Semantics

SL is the formal language used to define FIPA ACL's semantics. SL is a quantified, multi-modal logic with modal operators for *beliefs* (B), *desires* (D), *uncertain beliefs* (U), and *intentions* (or, *persistent goals*, PG). SL can represent propositions, objects, and actions. We can trace SL's origins to the work of Cohen and Levesque [8] but its current form is primarily based on the work [37]. A detailed description of SL, including its own semantics, is outside the scope of this article and can be found in the FIPA ACL specification.

In FIPA ACL the semantics of each communicative act are specified as sets of SL formulae that describe the act's feasibility preconditions and its rational effect. For a given CA a, the feasibility preconditions FP(a) describe the necessary conditions for the sender of the CA. That is, for an agent to properly perform the communicative act a by sending a particular message, the feasibility preconditions must hold for the sender. The agent is not obliged to perform a if FP(a) holds, but it can if it chooses. A communicative act's rational effect represents the effect that an agent can expect to occur as a result of performing the action; it also typically specifies conditions that should hold true of the recipient. The receiving agent is not required to ensure that the expected effect comes about and might indeed find it impossible. Thus, an agent can use its knowledge of the rational effect to plan what CA to perform, but it cannot assume that the rational effect will necessarily follow.

Conformance with the FIPA ACL means that when agent i sends CA a, the FP(a) for i must hold. The unguaranteed RE(a) is irrelevant to the conformance issue.

This introduction should be enough for a basic understanding of the example in Figure 3, which shows the specification of the communicative act inform, in which agent i informs agent j of content f. The content of inform is a proposition, and its meaning is that the sender informs the receiver that a given proposition is true. According to this semantics, the sending agent

1. holds that the proposition $B_i(f)$ is true ;
2. does not already believe that the receiver has any knowledge of the truth of the proposition $B_i(Bif_j(f) \vee Uif_j(f))$; and
3. intends that the receiving agent should also come to believe that the proposition is true (rational effect $B_j(f)$);

$< \mathbf{i}, \mathbf{inform}(\mathbf{j}, \mathbf{f}) >$
FP: $B_i(f) \wedge \neg B_i(Bif_j(f) \vee Uif_j(f))$
RE: $B_j(f)$

Fig. 3. FIPA ACL semantics for the communicative act inform. Agent i informs agent j of content f.

5 Comparing KQML and FIPA ACL

KQML and FIPA ACL are almost identical with respect to their basic concepts and the principles they observe. The two languages differ primarily in the details of their semantic frameworks. In one sense, this difference is substantial: because of the different semantic frameworks it would be impossible to come up with exact mappings or transformations between KQML performatives and their completely equivalent FIPA primitives, or vice versa. On the other hand, the ineluctable differences might be of little importance to many agents' programmers, if their agents are not true BDI agents.

Both languages assume a basic non-commitment to a reserved content language. However, in the FIPA ACL case, as we mentioned, an agent must have some limited understanding of SL to properly process a received message (as in the case of the request CA). The two languages have the same syntax. That is, a KQML message and a FIPA ACL message look syntactically identical -except, of course, in their different names for communication primitives. This is an important attribute of FIPA ACL. FIPA changed the language's original, Prolog-like syntax to match KQML's to facilitate the transition of KQML systems to FIPA ACL. A large part of making an agent system communication-ready is to provide code that will parse incoming messages, compose messages for transport, and channel them through the network using a lower- level network protocol. Identical syntaxes guarantee that this infrastructure will be the same regardless of the choice of ACL.

These encouraging thoughts do not apply to the semantics of the two languages. Following the KQML semantics described elsewhere, [22] [25] we can see that semantically the two languages differ at the level of what constitutes the semantic description: preconditions, postconditions, and completion conditions for KQML; feasibility preconditions and rational effect for FIPA ACL. They also differ at the level of the choice and definitions of the modalities they employ (the language used to describe agents' states). Although we can approximate the KQML primitives in FIPA's framework and vice versa, a complete and accurate translation is not, in general, possible. For example, to define a CA in FIPA ACL that approximates KQML's tell, we can replace f in the definition of inform with $B_i(f)$ (see Figure 3); still, the two definitions will not be semantically equivalent.

Another difference between the two ACLs is in their treatment of the registration and facilitation primitives. These primitives cover a range of important pragmatic issues, such as registering, updating registration information, and finding other agents that can be of assistance in processing requests. In KQML, these tasks are associated with performatives that the language treats as first-class objects. FIPA ACL, intended to be a purer ACL, does not consider these tasks CA's in their own right. Instead, it treats them as requests for action and defines a range of reserved actions that cover the registration and life-cycle tasks. In this approach, the reserved actions do not have formally defined specifications or semantics and are defined in terms of natural-language descriptions.

Many ACL users have expressed their desire that FIPA ACL include the facilitation primitives that they are accustomed to from KQML (broker, recommend, and recruit). Such user requests serve as a sobering reminder that to be practical, an ACL requires a careful mix of the theoretical and the pragmatic. FIPA is currently exploring the accommodation of facilitation primitives in FIPA ACL.

In theory, the similarity in basic assumptions and syntax among existing ACLs (under an assumption of shared naming and registration conventions) means that only the communication primitive-specific code should change according to the choice of ACL. Even then, much to the dismay of those defining ACL semantics, the implementers' intuitive understanding of the primitives might prevail over the concise semantic definitions [8]. So, unless an agent implements modalities (such as belief, desire, intention, and so on) following the particular agent theory that the semantic account suggests, the decision of which language to chose should be based on pragmatic concerns.

[8] When asking programmers about the development of the communication primitive-specific code, the author has often received responses that amount to *we read the natural language definition of the primitives.*

6 Agent Communication Languages: Beyond the Semantics

The emergence of FIPA ACL might have been a additional headache for implementers who must decide for themselves which one of the two ACLs to use. The largely semantic (and subtle) differences between the two ACLs pale in comparison to the more significant issues in terms of the practically useful incorporation of ACLs in agent systems.

In principle, any system that is to use KQML (or FIPA ACL, for that matter) must provide the following things:

- a suite of APIs that facilitate the composition, sending, and receiving of ACL messages;
- an infrastructure of services that assist agents with naming, registration, and basic facilitation services (finding other agents that can do things for your agent); and
- code for every reserved message type (performative or communicative act) that takes the action(s) prescribed by the semantics for the particular application; this code depends on the application language, the domain, and the details of the agent system using the ACL.

Ideally, a programmer should only have to provide item 3. Items 1 and 2 should be reusable components that the programmer integrates into the application code. Actually, the programmer should not even have to integrate the listed under item 2; they ought to exist as a continuous running service available to any new agent.

In practice, the craft of building ACL-capable agents is a bit more complex. First of all, the canonical ACL message syntax (both in FIPA ACL and KQML) further includes additional message parameters whose semantics go beyond that of the primitives. These parameters are unaccounted for in the deep semantics but are essential to the processing of an ACL message. For example: the ACL specification does not specify any conventions for a naming scheme and an associated namespace; or, there is nothing in the message that suggests the method of delivery (protocol). In other words, the ACL includes several pragmatic (i.e., operational) aspects, in addition to what is formally specified at the semantics layer. Such pragmatic aspects are necessary for parsing in and out of the ACL, i.e., digesting and composing well-formed ACL syntax (which is Lisp-like) to extract or insert message parameters or, for queuing (and de-queuing) ACL messages for delivery through TCP or some other network protocol. Further pragmatic issues include the conventions for finding agents and initiating interaction.

Although, in our view, these issues are actually outside of the ACL's scope they are fundamental to an agent's ability to speak an ACL. Research groups that designed and built agent systems had to establish their own conventions and choices for these pragmatic issues. Actually, the various APIs for KQML and FIPA ACL provide nothing (as expected) regarding the actual processing of

ACL messages (depending on the primitive), since respecting the deep semantics of the primitives is the responsibility of the application that makes use of those API's. Such API's today mainly take care of the parsing and queuing tasks mentioned above. Performing these tasks is what using KQML (or FIPA ACL, for that matter) has come to mean. For all intents and purposes, compliance with the ACL's specification means compliance with all these pragmatic conventions. These conventions are not part of the standard (to the extent that the ACL semantics is standardized) and the subtle (or not so subtle) discrepancies amongst their implementations account in large part for the situation today in which there is often a lack of interoperability between systems using the same ACL[9].

Such issues go beyond the specification of the ACL itself. After all, syntax and semantics suffice to fully specify any language, including an ACL. Still, without solutions to these questions, an ACL is only a useless artifact. FIPA was the first comprehensive effort to address collectively these matters; we outline some of its contributions in Setction 9. Next, we introduce the topics that have emerged as fundamental to the construction of interesting systems of communicating agents. Our opinion is that agent communication language standardization, refers to agent communication standardization, which includes issues and considerations that transcend the semantics. In the remaining of this article, we cover in more detail the current work on these issues.

Syntax and Encoding

The core semantics of an ACL is defined as the *deep* semantics (i.e., semantics in the sense of declarative knowledge representation) of its (communication) primitives. This semantics are expressed in some knowledge representation language: SL in the case of FIPA ACL. This semantics only takes into account the speaker, the hearer (in speech act terminology) and the content of the communicative act. The speaker, the hearer and the content correspond to the `:sender`, the `:receiver` and the `:content` of the syntactic representation of the ACL. The canonical syntactic form of the ACL message (for both KQML and FIPA ACL) is a Lisp-like ASCII sequence.

We advocate an abstract syntax for the ACL. Fixed elements of the abstract syntax are the ones that have a direct semantic equivalent (sender, receiver, performative, content). Messages might have multiple encodings, Java objects, Lisp-like ASCII, etc. We advocate XML as a more interoperable and flexible encoding and we elaborate these points in Section 8.

Services and Infrastructure

Every multiagent system that uses an ACL has a homegrown implementation of the APIs that we mentioned at the opening of this section. There are more than

[9] The differences in sets of primitives used and their intended meaning constitute a second-in-order interoperability barrier that is not confronted due to these more mundane “lower-level” obstacles.

a handful of APIs written in Java, for Java agents and each provides its own infrastructure of basic services. Such API's provide the means to compose well-formed ACL messages, to be able to parse them, to send and receive them over the network and to have some scheme for naming agents, etc. We do not expect a one-size-fits-all solution to these problems. We advocate WWW-like naming schemes and multiple encodings (or, XML encodings, as we hope) as means for reducing the overhead of building an agent that can interact with other, existing agents. An XML encoding will help, for example, with the overhead of composing and parsing because there are XML-processing tools. For facilitators, name servers, *etc.*, things will remain in flux because solutions and approaches are application and domain dependant, but it would be much easier to develop such services if we can build on top of a WWW- like infrastructure of agent naming and message encoding.

Conversations

As we mentioned earlier, the original KQML specification suggested an implicit sequencing of messages in agent interactions. First in [23] and later in [27] [5] [11] the idea of conversations for communicating agents that use an ACL, was introduced and further explored. Conversations mark a shift from individual messages to sequences (exchanges) that agents engage in order to perform certain tasks. The emphasis shifts from the agent's internals to the agent's behavioral patterns. In that sense they might be more useful than the semantics in software development. The body of work in the area has been impressive enough to warrant at least on workshop on conversations for communication agents [10].

ACL's and the WWW

There was no WWW when KQML first appeared (there were no agents, either, for that matter). KQML and FIPA ACL have evolved at a considerable distance from the mainstream of Internet technologies and standards. No Internet standardization organization has ACL's in their agenda. With the exception of the Artimis project (France Telecom), no major industry player has committed major resources to depend upon, or to develop, ACL's, although there are some plans for future work that will take advantage of FIPA technologies, as they become available. At the same time the WWW is a huge repository of information and agents are almost always referred to in conjunction with the WWW. ACL's are driving a great part of the agent work (FIPA ACL is the centerpiece of the FIPA effort); it is thus reasonable to suggest that ACL work ought to integrate easily with the WWW and to be able to leverage WWW tools and infrastructure. Some first step towards integrating ACL work into the WWW include: abstract syntax for ACL messages, domain specific content languages, a XML encoding for the messages and the content. And all that, need not be at the exclusion of any type of agents, e.g., mobile agents. Of particular interest are

[10] In the Agents-99 conference.

efforts to create content languages that are suitable *Knowledge Representations* for WWW content such as the DARPA Agent Modelling Language (DAML [11]) and RuleML [12].

7 Agent Communication Languages and Conversations

An Agent Communication Language provides agents with a means to exchange information and knowledge. ACLs, such as KQML or FIPA ACL, are languages of propositional attitudes. ACLs are intended to be above the layer of mechanisms such as RPC or RMI because: (1) they handle propositions, rules and actions instead of simple objects (with no semantics associated with them), and (2) the ACL message describes a desired state in a declarative language, rather than a procedure or method. But ACLs by no mean cover the entire spectrum of what agents may want to exchange. More complex *objects* can and should be exchanged between agents, such as shared plans and goals, or even shared experiences and long-term strategies.

As discussed, the ACL itself defines the types of messages (and their meaning) that agents may exchange. Agents though, do not just engage in single message exchanges but they have *conversations*, *i.e.*, task-oriented, shared sequences of messages that they observe, in order to accomplish specific tasks, such as a negotiation or an auction. At the same time, some higher-level conceptualization of the agent's strategies and behaviors drives the agent's communicative (and non-communicative) behavior. When an agent sends a message, it has expectations about how the recipient will respond to the message. Those expectations are not encoded in the message itself; a higher-level structure must be used to encode them. The need for such conversation policies is increasingly recognized by both the KQML [27] and the FIPA communities [15,12].

A conversation is a pattern of message exchange that two (or more) agents agree to follow in communicating with one another. In effect, a conversation is a pre-arranged coordination protocol, A conversation lends context to the sending and receipt of messages, facilitating interpretation that is more meaningful.

Although, conversations have become part of many infrastructures for ACL-speaking agents, relatively little work has been devoted to the problem of conversation specification and implementation. For conversations to be used for agent coordination, the following three issues require attention:

1. Conversation specification: How can conversations best be described so that they are accessible both to people and to machines?
2. Conversation sharing: How can an agent use a specification standard to describe the conversations in which it is willing to engage, and to learn what conversations are supported by other agents?
3. Conversation aggregation: How can sets of conversations be used as agent API's to describe classes of capabilities that define a particular service?

[11] http://www.daml.org
[12] http://www.dfki.de/ruleml

Well-defined, sharable conversation protocols, with testable, desirable properties, can be used to coordinate agents that attempt to accomplish specific tasks. Typically, a conversation protocol is associated with a specific task, such as *registration*, or a particular type of a *negotiation*. Agents adhering to the same conversation protocol, can coordinate their communicative actions as they attempt to accomplish the task suggested by the conversation protocol. Such a coordination is akin to a scripted interaction, with specific properties, rather than coordinated action resulting from a deep understanding of the domain and the task at hand. Nevertheless, it can be effective for tasks that can be adequately described in the form of possible sequences of communicative interactions.

A specification of a conversation that could be shared among agents must contain several kinds of information about the conversation and about the agents that will use it. First, the sequence of messages must be specified. Traditionally, *deterministic finite-state automata* (DFA's) have been used for this purpose; DFA's can express a variety of behaviors while remaining conceptually simple. For more sophisticated interactions, however, it is desirable to use a formalism with more support for concurrency and verification. Next, the set of roles that agents engaging in a conversation may play must be enumerated, and the constraints and dependencies between individual messages need to be captured. Many conversations will be dialogues, and will specify just two roles; however conversations with more than two roles are equally important, representing the coordination of communication among several agents in pursuit of a single common goal. For some conversations, the set of participants may change during the course of the interaction.

These capabilities will allow the easy specification of individual conversations. To develop systems of conversations though, developers must have the ability to extend existing conversations through specialization and composition. Specialization is the ability to create new versions of a conversation that are more detailed than the original version; it is akin to the idea of subclassing in an object-oriented language. Composition is the ability to combine two conversations into a new, compound conversation. Development of these two capabilities will entail the creation of syntax for expressing a new conversation in terms of existing conversations, and for linking the appropriate pieces of the component conversations.

The set of conversations in which an agent will participate defines an interface to that agent. Thus, standardized sets of conversations can serve as abstract agent interfaces (AAIs), in much the same way that standardized sets of function calls or method invocations serve as APIs in the traditional approach to system-building. That is, an interface to a particular class of service can be specified by identifying a collection of one or more conversations in which the provider of such a service agrees to participate. Any agent that wishes to provide this class of service need only implement the appropriate set of conversations.

Implementing and expressing conversations is not a new idea. As early as 1986, Winograd and Flores [40] used state transition diagrams to describe conversations. The COOL system [3] has perhaps the most detailed current FSM-based model to describe agent conversations. Each arc in a COOL state transition diagram represents a message transmission, a message receipt, or both. One consequence of this policy is that two different agents must use different automata to engage in the same conversation. Other conversation models have been developed, using various approaches. Extended FSM models, which, like COOL, focus more on expressivity than adherence to a model, include Kuwabara et al. [21], who add inheritance to conversations, Wagner et al. [39], and Elio and Haddadi [13], who defines a multi-level state machine, or Abstract Task Model (ATM). A few others have chosen to stay within the bounds of a DFA, such as Chauhan [7], who uses COOL as the basis for her multi-agent development system [13], Nodine and Unruh [32], and Pitt and Mamdani [35], who uses DFAs to specify protocols for BDI agents. Also using automata, Martin et al. [29] employs Push-Down Transducers (PDT). Lin et al. [28] and Cost et al. [10] demonstrate the use of CPNs, and Moore [30] applies state charts. Parunak [33] introduces Dooley Graphs. Bradshaw [4] introduces the notion of a conversation suite as a collection of commonly-used conversations known by many agents. Labrou [22] uses definite clause grammars to specify conversations.

While each of these works makes contributions to our general understanding of conversations, more work needs to be done to facilitate the sharing and use of conversation policies by agents. FIPA has contributed to the overall effort by providing specifications for a number of interesting conversation protocols.

8 Agent Communication and the WWW

A number of researchers within and outside FIPA have suggested that ACL messages ought to be encoded in XML, in their entirety, i.e., both the message layer and the content layer should be in XML.

XML is a language for creating markup languages that describe data. XML is a machine-readable and application-independent encoding of a *document*, e.g., of a FIPA ACL message including its content. In contrast to HTML which describes document structure and visual presentation, XML describes data in a human-readable format with no indication of how the data is to be displayed. It is a database-neutral and device-neutral format; data marked up in XML can be targeted to different devices using, for example, *eXtensible Style Language* (XSL). The XML source by itself is not primarily intended directly for human viewing, though it is human-understandable. Rather, the XML is rendered using standard available XML-world tools, then browsed, e.g., using standard Web browsers or specialized other browsers/editors. One leading method for rendering is via XSL, in which one specifies a stylesheet.

XML is a meta-language used to define other domain- or industry-specific languages. To construct a XML language (also called a *vocabulary*), one supplies a specific *Document Type Definition* (DTD), which is essentially a context-free grammar like the Extended BNF (Backus Naur Form) used to describe computer

[13] More recent work with this project, JAFMAS, explores conversion of policies to standard Petri Nets for analysis [16].

languages. In other words, a DTD provides the rules that define the elements and structure of the new language. For example, if we want to describe employee records, we would define a DTD which states that the <NAME> element consists of three other elements called <FIRST>, <MIDDLE>, and <LAST>, in that order. The DTD would also indicate if any of the nested elements is optional, can be repeated, and/or has a default value. Any browser (or application) having an XML parser could interpret the employee document instance by learning the rules defined by the DTD.

Elsewhere, [19] we have defined ACML (Agent Communication Markup Language), a XML language for describing FIPA ACL message. The deep semantics of the communication primitives in ACML is simply taken to be the same as previously. This semantics is not affected by encoding in XML instead of the previous ASCII; it is defined independently of the choice of syntactic encoding.

Encoding ACL messages in XML offers some advantages that we believe are potentially quite significant.

The XML-encoding is easier to develop parsers for than the Lisp-like encoding. The XML markup provides parsing information more directly. One can use the off-the-shelf tools for parsing XML, instead of writing customized parsers to parse the ACL messages. A change or an enhancement of the ACL syntax does not have to result to a re-writing of the parser. As long as such changes are reflected in the ACL DTD, the XML parser will still be able to handle the XML-encoded ACL message. In short, a significant advantage is that the process of developing or maintaining a parser is much simplified.

More generally, XML-ifying makes ACL more WWW-friendly, which facilitates Software Engineering of agents. Agent development ought to take advantage and build on what the WWW has to offer as a software development environment. XML parsing technology is only one example. Using XML will facilitate the practical integration with a variety of Web technologies. For example, an issue that has been raised in the ACL community is that of addressing security issues, e.g. authentication of agents' identities and encryption of ACL messages, at the ACL layer. The WWW solution is to use certificates and SSL. Using the same approach for agent security considerations seems much simpler and more intuitive than further overloading ACL messages and the ACL infrastructure to accommodate such a task.

As we mentioned earlier, the operational semantics of the pragmatic aspects of ACL can differ subtly between implementations or uses, and there is today a problem practically of interoperability. XML can help with these pragmatics, by riding on standard WWW-world technologies: to facilitate the engineering, and as a by-product to help standardize the operational semantics, thereby helping make interoperability really happen.

Because XML incorporates links into the ACL message, this takes a significant step toward addressing the problem (or representational layer) of specifying and sharing the ontologies used in an ACL message's content. The values of the ACL parameters are not tokens anymore, but links that can point to objects and/or definitions. Although the ontology slot has been present since the inception of ACLs, the ACL community has not been very clear on how this information is to be used by the agent. This vagueness, further compounded by the scarcity of published ontologies, can be addressed by interfacing the ACL message to the knowledge repository that is the WWW.

More generally, links may be useful for a variety of other purposes. For example, the receiver parameter might have a link to network location that provides information about the agent's identity: e.g., its owner, contact and administrative information, communication primitives that the agent understands, network protocols and ports at which it can receive messages, conversation protocols it understands, etc.. This type of information is necessary for a establishing an extended interaction with another agent and has to somehow be available to an agent's potential interlocutors. The same argument can be made about the other message parameters.

9 FIPA: Services and Infrastructure

FIPA's approach on the ACL specification is somewhat different than that of KQML, partly because FIPA is an organized body with the authority to take action on its own specifications and the KQML community is a loose network of researchers.

The initial intent was to provide an ACL specification that only covered syntax and semantics of the primitives, carefully avoiding addressing any of the pragmatic considerations at the ACL specification level. Most of the pragmatic considerations were delegated to a different group, charged with Agent Management. The boundaries between the two were often challenged, but with few exceptions (such as conversation specifications been introduced as part of the ACL specification document) the divide persevered. The conflict relates to striving for a balance between a semantically (and theoretically) *pure* ACL specification and the need for a *practical* ACL, from the system-building point of view.

In recognition of the importance of conversations FIPA provides a number of pre-specified protocols. Additionally, FIPA has addressed some of the other issues we mentioned previously, in its Agent Management Transport and Agent Management specifications [14].

The FIPA Agent Message Transport specifications deal with the delivery and representation of messages across different network transport protocols, including wireline and wireless environments. The agent message transport reference model provides facilities for: (1) general support for a Message Transport Service within an agent platform, (2) guidelines for using specific Message Transport Protocols, such as IIOP, HTTP and WAP and (3) support for encodings that are suitable for each transport protocol, such as an XML encoding for HTTP and a bit-efficient encoding for WAP.

The FIPA Agent Management Specification provides the framework within which FIPA agents exist and operate. It establishes the logical reference model

[14] See [6] for a recent summary of these specifications.

for the creation, registration, location, communication, migration and retirement of agents. The reference model describes the primitives and ontologies necessary to support the following services in an agent platform: (1) white pages, such as agent location, naming and control access services, (2) yellow pages, such as service location and registration services, which are provided by the *Directory Facilitator*.

In the FIPA model agents belong to one or more agent platforms which provide basic services in a way consistent with the above specifications. Sixteen FIPA platforms have been implemented by diverse companies, four of these are freely accessible under open source, that support these specifications. These FIPA platforms have been distributed and tested in large-scale projects, which collectively have been downloaded several thousands of times. For the latest information on FIPA specifications and available FIPA-compliant platforms, the reader is encouraged to check the FIPA homepage[15].

10 Conclusions

When Knowledge Query and Manipulation Language (KQML) first appeared 10 years ago, it was not an Agent Communication Language (ACL), the term *agents* did not refer to the same kind of entity it refers to today and the KQML specification was a little more than a document combining natural language and syntactic sugaring. Over a period of time KQML has come to define the concept of an ACL and in the process the ACL has become the centerpiece of a large category of agent systems. Almost inevitably an ACL has became a loosely-defined concept that encompasses a variety of issues which may or may not be ACL-relevant depending on one's point of view. The more *conservative* viewpoint advocates that semantics is the one and only real issue. Agent development often suggests though that semantics is the least important concern when one actually builds an agent system. The emergence of FIPA ACL was touted as an attempt for a *cleaner*, *purer* ACL with well-defined semantics. But aside from the inherent limitations of current ACL semantic approaches, the efforts of many researchers to develop multi-agent systems have brought to the foreground issues and considerations that are at least as important as the semantics for interoperable agent systems; FIPA faced the same problems that the KQML community had been unable to put to rest. The naive (in hindsight) early concept of KQML messages as speech-act-like message types, expressed as ASCII strings, with a LISP-like syntax, that are transported over TCP/IP connections, and aimed at knowledge and information exchange between software systems that are viewed as Virtual Knowledge Bases, exhibited its limitations a long time ago.

Standardization, though, revolves mainly around usability. How to use the semantics in order to develop agent systems that communicate and interoperate using a ACL? Are conversation protocols a more useful tool for agent development than the semantics? Where are the basic services that the ACL presumes

[15] http://www.fipa.org

to exist? Do ACL messages have to be ASCII strings encoded in an AI-ish syntax? What do agents actually ”talk” about? ACL messages are expected to be opaque to the content language expressions they ”carry” but this does not answer the question of which content languages are useful for applications. Is there any connection between agents, ACL’s and the World Wide Web? These are the questions we investigated. To some of them we provided (partial) answers and examined the current work, to others we suggested possible avenues of research; our persistent goal, though, was to finally get past the semantics and come to terms with a broader notion of agent communication standardization.

References

1. ARPA Knowledge Sharing Initiative. Specification of the KQML agent-communication language. ARPA Knowledge Sharing Initiative, External Interfaces Working Group, July 1993.
2. J. L. Austin. *How To Do Things With Words*. Harvard University Press, second edition, 1962, 1975.
3. Mihai Barbuceanu and Mark S. Fox. COOL: A language for describing coordination in multiagent systems. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 17–25, San Francisco, CA, 1995. MIT Press.
4. Jeffrey M. Bradshaw. KAoS: An open agent architecture supporting reuse, interoperability, and extensibility. In *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1996.
5. Jeffrey M. Bradshaw, Stuart Dutfield, Pete Benoit, and John D. Woolley. Kaos: Toward an industrial-strength open agent architecture. In Jeffrey M. Bradshaw, editor, *Software Agents*. AAAI/MIT Press, 1997.
6. Bernard Burg, Jonathan Dale, and Steven Willmott. Open standards and open source for agent-based systems. *AgentLink*, January 2001.
7. Deepika Chauhan. JAFMAS: A Java-based agent framework for multiagent systems development and implementation. Master’s thesis, ECECS Department, University of Cincinnati, 1997.
8. Philip R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–361, 1990.
9. Philip R. Cohen and H.J. Levesque. Communicative actions for artificial agents. In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS’95)*. AAAI Press, June 1995.
10. R. Scott Cost, Ye Chen, Tim Finin, Yannis Labrou, and Yun Peng. Modeling agent conversations with colored petri nets. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 59–66, Seattle, Washington, May 1999.
11. R. Scott Cost, Tim Finin, Yannis Labrou, Xiaocheng Luan, Yun Peng, Ian Soboroff, James Mayfield, and Akram Boughannam. Jackal: A java-based tool for agent development. In *Working Notes of the Workshop on Tools for Developing Agents (AAAI Technical Report)*, Madison, WI, 1998.
12. Ian Dickinson. Agent standards. Technical report, Foundation for Intelligent Physical Agents, October 1997.

13. Renée Elio and Afsaneh Haddadi. On abstract task models and conversation policies. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 89–98, Seattle, Washington, May 1999.
14. Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua server: A tool for collaborative ontology construction. In *KAW96*, November 1996.
15. FIPA. FIPA 97 specification part 2: Agent communication language. Technical report, FIPA - Foundation for Intelligent Physical Agents, October 1997.
16. Alan Galan and Albert Baker. Multi-agent communications in JAFMAS. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 67–70, Seattle, Washington, May 1999.
17. Michael Genesereth and Richard Fikes. Knowledge Interchange Format, version 3.0 reference manual. Technical report, Computer Science Department, Stanford University, June 1992.
18. Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, 1994.
19. B. Grosof and Y. Labrou. An approach to using xml and a rule-based content language with an agent communication language, 1999.
20. Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 2:199–220, 1993.
21. Kazuhiro Kuwabara, Toru Ishida, and Nobuyasu Osato. AgenTalk: Describing multiagent coordination protocols with inheritance. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '95)*, pages 460–465, 1995.
22. Yannis Labrou. *Semantics for an Agent Communication Language.* PhD thesis, University of Maryland, Baltimore County, August 1996.
23. Yannis Labrou and Tim Finin. A semantics approach for KQML-a general purpose communication language for software agents. In *3rd International Conference on Information and Knowledge Management*, November 1994.
24. Yannis Labrou and Tim Finin. A proposal for a new kqml specification. Technical Report Technical Report TR-CS-97-03, University of Maryland Baltimore County, 1997.
25. Yannis Labrou and Tim Finin. Semantics and conversations for an agent communication language. In Michael Huhns and Munindar Singh, editors, *Readings in Agents*. Morgan Kaufmann, 1997. Reprint of a paper from the Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997 (IJCAI-97).
26. Yannis Labrou, Tim Finin, and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, / 1999.
27. Yannis Labrou and Timothy Finin. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, August 1997.
28. Fuhua Lin, Douglas H. Norrie, Weiming Shen, and Rob Kremer. Schema-based approach to specifying conversation policies. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Third International Conference on Autonomous Agents*, pages 71–78, Seattle, Washington, May 1999.
29. Francisco Martin, Enric Plaza, and Juan Rodríguez-Aguilar. Conversation protocols: Modeling and implementing conversations in agent-based systems. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 49–58, Seattle, Washington, May 1999.

30. Scott Moore. On conversation policies and the need for exceptions. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 19–28, Seattle, Washington, May 1999.
31. R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall 1991.
32. M. H. Nodine and A. Unruh. Facilitating open communication in agent systems: the InfoSleuth infrastructure. In Michael Wooldridge, Munindar Singh, and Anand Rao, editors, *Intelligent Agents Volume IV – Proceedings of the 1997 Workshop on Agent Theories, Architectures and Languages*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 281–295. Springer-Verlag, Berlin, 1997.
33. H. Van Dyke Parunak. Visualizing agent conversations: Using enhanced dooley graphs for agent design and analysis. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS '96)*, 1996.
34. Ramesh S. Patil, Richard E. Fikes, Peter F. Patel-Schneider, Don McKay, Tim Finin, Thomas Gruber, and Robert Neches. The darpa knowledge sharing effort: Progress report. In Michael Huhns and Munindar Singh, editors, *Readings in Agents.* Morgan Kaufmann Publishers, 1997. (reprint of KR-92 paper).
35. Jeremy Pitt and Abe Mamdani. Communication protocols in multi-agent systems. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 39–48, Seattle, Washington, May 1999.
36. Jeremy Pitt and Abe Mamdani. A protocol-based semantics for an agent communication language. In *IJCAI*, pages 486–491, 1999.
37. M.D. Sadek. A study in the logic of intention. In *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 462–473, Cambridge, MA, 1992.
38. Ira A. Smith and Philip R. Cohen. Toward a semantics for an agent communications language based on speech-acts. In *Proceedings of the 13th National Conference on Artificial Intelligence.* AAAI/MIT Press, August 1996.
39. Thomas Wagner, Brett Benyo, Victor Lesser, and Ping Xuan. Investigating interactions between agent conversations and agent control components. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 79–88, Seattle, Washington, May 1999.
40. Terry Winograd and Fernando Flores. *Understanding Computers and Cognition.* Addison-Wesley, 1986.

Standardizing Agent Interoperability: The FIPA Approach

Stefan Poslad[1] and Patricia Charlton[2]

[1] Department of Electronic Engineering, Queen Mary, University of London,
Mile End Road, London, E1 4NS.
stefan.poslad@elec.qmul.ac.uk
[2] Centre de Recherche de Motorola-Paris
Espace Technologique – Commune de Saint Aubin
91193 Gif-sur-Yvette, France.
patricia.charlton@crm.mot.com

Abstract. A prolific number of different Multi-Agent Systems (MAS) and associated applications have been developed in numerous research institutes and industrial laboratories world-wide. Perhaps the most important barrier to MAS making a successful transition from this research environment towards widespread adoption for consumer products and businesses, is the lack of interoperability between heterogeneous MA Systems. In 1996, the **F**oundation for **I**ntelligent **P**hysical **A**gents (FIPA) was formed to provide a forum for developing specifications for agent systems. Since its formation, FIPA has increasingly focussed more on standardizing (multi-agent system) agent interoperability. As a result, it is often said that FIPA really stands for the **F**oundation for **I**ntero**P**erable **A**gents. In this article, we discuss both technical and scientific issues in defining standards for interoperability between agents in different MA systems with a particular focus on the FIPA agent interoperability standards.

1 Introduction

In 1996, FIPA, The Foundation for Intelligent Physical Agents [1], a forum of international companies with a strong focus in the telecommunication industry, was formed to promote the uptake of software agents in businesses at large. It was originally intended that specifications and standards, encompassing both hardware (physical) agents such as robots and software agents, would be developed hence the use of the term physical in the FIPA full name. However, as FIPA progressed, the interest of the forum focused increasingly on software rather than hardware agents. A second key focus was on specifying communication and interoperability between agents rather than specifying how agents processed and reasoned about the information they received. That is, FIPA concentrated on standardising 'external intelligence' or rich interoperability rather than on 'internal intelligence' or reasoning. FIPA's official mission statement is: "The promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent agent systems in modern commercial and industrial settings." Hence, it is said that perhaps FIPA more accurately stands for "*F*oundation for *I*ntero*P*erable *A*gents" or "Foundation for Interoperable Peer-to-peer Agents.

FIPA produced its first set of seven specifications in 1997. This set of specifications included: an agent architecture referred to as the FIPA 1997 agent platform; an agent communication language and some applications such as travel assistance, network management, audio-visual entertainment and personal assistance In subsequent years, further specifications were added, the architecture and agent communication language have been refined, the specification process has been formalized and there is more focus on abstractions and instantiations of the abstractions for a heterogeneous world.

The remainder of the article is structured as follows. We first (in the following two sections, 2 and 3) try to give some insight into why and how FIPA has developed specifications in a particular way. Then we summarise the specifications (section 4). Then, in section 5, issues governing the implementation and use of the specifications are considered. Finally, a conclusion about FIPA's work is given.

2 Modelling a Richer Type of (Agent-Based) Interoperability

Multi-agent architectures are distributed systems in which each agent is: able to act autonomously, able to reason and make decisions about the environment in which it is embedded, and able to interact with other agents. The distributed and autonomous nature of agent systems potentially supports flexibility and robustness in system operation and organization, particularly when these are coupled with dynamic system behaviour. Highlighting "agent systems" as autonomous helps to highlight the usefulness of agents in solving complex real world problems.

As the area covering agents and agent technology is a heterogeneous body of research and development, there are several dimensions in which one can characterise agent software. Frankin and Graesser [2] identify the following dimensions to characterise agent software: reactivity, autonomy, proactivity, responsibility, continuity, interactivity, adaptability, rationally, cooperativity and robustness. The above properties are considered generally useful for an agent. However, it should be noted that conditions for agent-hood are not necessarily environmentally determined. The environment will determine which properties favour success and which cause failure. Thus, there are of course other properties that are not critical to a general notion of agent-hood, but that can be used as a basis for the classification of agents. One example is agent mobility: an agent that is able to transport itself from one site to another across a network. Mobility however does become a necessary condition of agent-hood in certain environments, those where hard real-time constraints exist at remote servers, where large volumes of data must be processed, where processing resources of agent servers are over-utilised, and where the communication channel is costly or hostile.

For the purposes of our discussion of the FIPA specifications, we focus primarily on the autonomy and interactivity dimensions of agents. In order to usefully exploit both the autonomy and interactivity, engineering solutions using agents, the partitioning of the domain or problem by explicitly abstracting certain characteristics of the domain and associating these characteristics with an agent's role is required.

At its core an organisational approach [3], rather than a top-down or bottom-up approach. There are several organisational-related aspects that are important factors influencing the identification and characterisation of agents. Some of these aspects are:

- an organisation can be divided into sub-groups having a cohesion that makes it natural to associate an agent with a specific sub-group.
- within a group, certain services should be provided, and services that to a large degree share domain knowledge can be realised by a single agent.
- a larger group might be partitioned into separate responsibilities, and this can cause separation of services into different agents for each responsibility.
- a group can be governed by a set of rules or policies (constraints) identifying what are legal states and behaviours, and agents should exhibit a behaviour that conforms to these rules.

Hence, to solve distributed complex problems using multiple, autonomous, communicative agents, we partition the problem into sub-problems and map these to one or more organisations of agent groups with a common behaviour and then to agents with individual roles that co-operate with other agents in the same group or in different groups.

There are two main ways that multi-agent systems can be modelled or expressed: they can be designed using descriptive frameworks, accompanied by some prescriptive element in order to constrain the behaviour or they can be modelled using a formal logical framework, whose behaviour is well defined and can be verified. The FIPA specifications contain elements of both these approaches. It is important to consider these because the definition of the model will affect how different implementations of the model can interoperate. These two approaches are now examined in more detail in turn.

2.1 FIPA Agent Specifications as Descriptive Models

There are many descriptive frameworks that describe and classify agent behaviour, for example, [2], [4] and [5]. Descriptive system frameworks can contain several different complementary views of the same system: a functional or behavioural model, a data or information model, an organisational model and an interaction or operational model.

The functional view or model defines the properties and behaviours of agents using informal descriptions. The functional view (and the operational and organisational view) can also define prescriptive elements: rules such as pre and post-conditions that determine when the function can and cannot operate. An interaction or operational model defines the sequences of functions that define the (normal) operation of the system. The data or informational model expresses the structure of the information for exchange and for persistence. The organisational model defines how parts of an agent are structured, how different types of agent are related and how instances of agents can form (organisational) groups that offer higher-level abstractions. These models can have different levels of granularity.

The FIPA specifications use all of these four types of model to define multi-agent systems. A top-level organisational model for FIPA agents is given in Figure 1. This is slightly adapted from that in the FIPA Abstract Architecture specification [6].

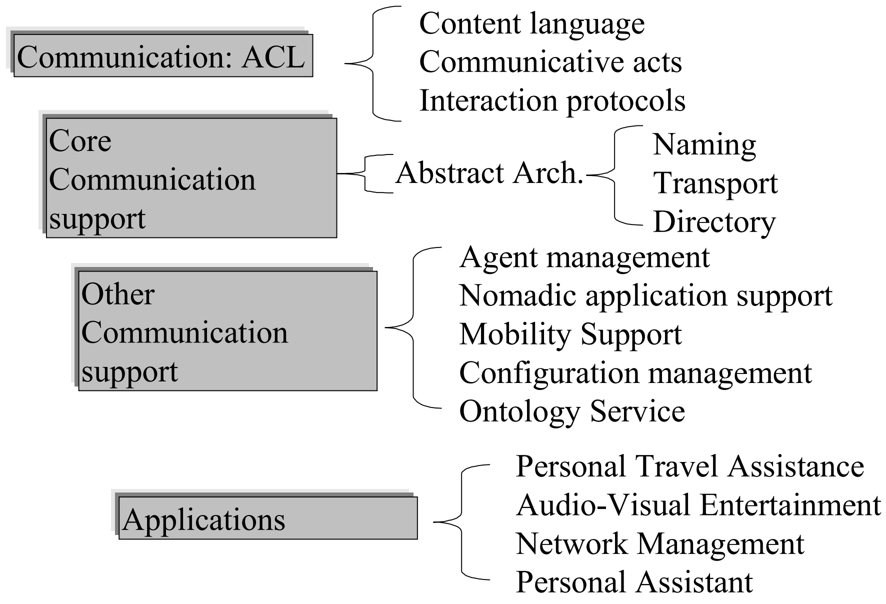


Fig. 1. A high-level organisational model of FIPA multi-agent systems

The functional model describes the behaviour of an agent component, for example, the following extract describes the function of the directory service as described in [6]. "The basic role of the directory-service function is to provide a location where agents register directory-entries. Other agents can search the directory-entries to find agents with which they wish to interact. The directory-entry is a key-value-tuple consisting of at least the following two key-value-pairs…In addition the directory-entry may contain other descriptive attributes, such as the services offered by the agent, cost associated with using the agent, restrictions on using the agent, etc…"

An example of a FIPA informational model is given in Table 1 and is taken from the FIPA transport specification [7]. This defines the header or wrapper that is added to ACL messages when they are transported between agents.

An example of a FIPA operational model is given in Figure 2, and is taken from the FIPA interaction protocol specification such as the FIPA-Request-Protocol [8]. This specifies a dialogue for one agent to request another to perform some action, and the receiving agent to perform the action or, to reply in some way that it cannot. The dialogue is represented in an agent extension of UML (Unified Modelling Language) called AUML or Agent UML [9]. This extension adds support for concurrent threads of interaction and the notion of an agent role. Pitt et al [10] have proposed a formal semantics for AUML.

Frame	`envelope`		
Ontology	`FIPA-Agent-Management`		
Parameter	**Description**	**Presence**	**Type**
`to`	This contains the names of the primary recipients of the message.	Mandatory	Sequence of `agent-identifier`
`from`	This is the name of the agent who actually sent the message.	Mandatory	`agent-identifier`
`comments`	This is a comment in the message envelope.	Optional	`String`
`acl-representation`	This is the name of the syntax representation of the message body.	Mandatory	`String`

Table 1. A FIPA informational model: fragment of the header for ACL messages.

Inherent in dialogues are constraints on the order of the messages in the dialogue, the messages can only occur in a certain order during normal interoperation between the sender and receiver. This is an example of a prescription. Prescription constrains the operation of the system to better support system management and interoperability. Prescription expresses the conditions under which elements can be normally be used. If these conditions are false, then either errors must be raised and handled or the system operation is suspended until the conditions are true again. Embedded in the specifications are many prescriptive statements or rules that to constrain elements of behaviour of FIPA agent systems.

For example, here are examples of prescriptive constraints from the FIPA agent management specification [12]:

- "An agent must have at least one owner .."
- "A Directory Facilitator (DF) is a mandatory component of the AP (Agent Platform) .."

The following rules are adopted to select the appropriate communicative act that will be returned when a management action causes an exception:

- "If the communicative act is not understood by the receiving agent, then the replied communicative act is not-understood."
- "If the requested action is not supported by the receiving agent, then the communicative act is refuse."

To summarise, the FIPA specifications are expressed using different types of descriptive models with various (informal) prescriptive rules to constrain the operation, behaviour and functions of FIPA agents.

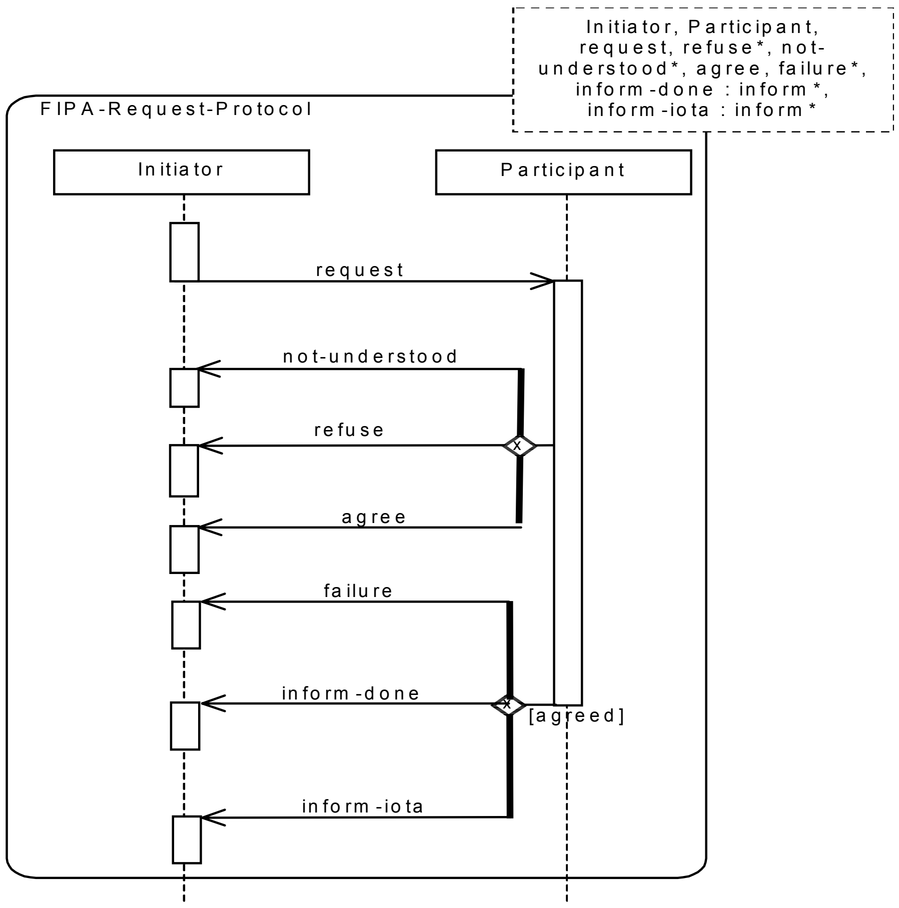


Fig. 2. A FIPA interaction model: the FIPA-Request Interaction Protocol

2.2 FIPA Agent Specifications as Formal Models

An alternative approach, to employing some kind of descriptive framework to define agent behaviour and in particular agent communication, is to use a more formal model to govern the agent behaviour. For example, the semantics of FIPA agent communication is (partially) underpinned by a formal logic. The FIPA ACL has formal semantics that are defined using a multi-modal (BDI-type) logic called Semantic Language or SL. The semantic model maps each agent message type (speech act) to an SL-formula that defines constraints that the sender must satisfy called the feasibility conditions. FIPA also maps each message to an SL formula that defines the rational effect of the action.

Putting aside theoretical issues such as whether or not such a formal model is well-defined or well-grounded, we could compute when each message is sent that it obeys the feasibility and rational effect conditions and hence we could verify that the message conforms to the FIPA formal-logic specification. This allows standard axiomatic proofs to determine and ascertain the behaviour of each aspect of the system. For example, if we have axiomatic proofs to describe the transmission of messages between a sender and a receiver and for describing the effect these messages cause at the receiver and the sender then we can ascertain when we have the correct behaviour for successfully transmitting a message. This would theoretically make maintaining and explaining the causes of interoperability easier.

2.3 Descriptive vs. Formal Models

When considering the form of specifications for supporting agent interoperability, there are persuasive arguments for using descriptive approaches, for using formalisms or for using a hybrid approach.

There are many perceived problems with the current formalism that underpins the FIPA ACL - SL. Firstly, the formalisms within the FIPA specifications are only used to model (simpler?) parts, such as the state before and after sending a particular speech act, of agent communication. For example, it does not explain how dialogues work, how agent systems can be managed and maintained or what do if partial failures occur in a distributed system. Secondly, various researchers have argued that the particular semantic logic used by FIPA is too limited to support (the various kinds of) agent interoperability. For example, Wooldridge [13] argues that the SL formalism does not lead to proofs that can be easily automated or computed – SL is "ungrounded". It is also generally accepted that intentional semantics are problematic within a distributed environment [14] hence the SL semantics for the rational effect, at the receiver, are not generally observed by many FIPA agents in practice. It is also proposed that different semantics are needed to better support agreements and observable commitments [10].

Thirdly, many particular logics may not be expressive enough, in practice, to capture the complexity of the real world; to be computationally tractable in a timely fashion and to be verifiable in a non-deterministic world or to function in a dynamic, extensible and open system environment.

There are limitations to a purely descriptive approach to specification even if that approach is complemented with prescriptive elements. It is simply not enough for specifications to gain widespread support and become a de facto standard. To gain, and maintain acceptance, particularly for sensitive applications such as electronic commerce, it must be possible to determine whether or not any system that claims to conforms to an ACL standard actually does so. An ACL standard is verifiable if it enjoys this property [13]. Traditionally, the way distributed systems are modelled, using descriptive systems, to show conformance to a standard that is specified in some descriptive framework, is to show that the actual output of the system, given some predefined input, equals the expected output. The weakness of this approach is that conformance, at points other than the defined conformance points, is undefined. A second complimentary approach is to define pre and post-conditions for the computation to determine when a function should be called and what the result should be after the function has been triggered. This enables the (correct) behaviour of the function to be verified (according to the computation conditions).

Both descriptive and formal approaches to specifications have strengths and weaknesses. If FIPA agents are to be used in domains such as eCommerce then verifiable models of operation and for interoperation are essential. The use of both descriptive models with prescriptive elements and formal, computationally grounded models of operation are important.

3 Standards for Multi-agent System Interoperability

In a heterogeneous world, concurrent distributed development has lead to many types of multi-agent systems that are islands of functionality – agents on different types of platform are unable to interoperate with each other. Agents from different vendors are likely to use different types of messages and message formats and the meaning and interpretation of the content is likely to differ. The driving force for interoperability is partly the customer who strives for simplicity and universality when accessing multiple services, and partly producers who often need to act in unison to obtain a critical mass for a sustainable customer-base.

Early adopters, who produce new technology and services, tend to be wary where there is no commonly agreed standard for interoperability and suspicious of standards that lack the support of a large consortium of companies. The standardization process helps shift the emphasis from the development of the infrastructure to the use of the infrastructure.

The main driving force for de facto standards from multiple-vendor forums is to seed the market and attain a critical mass of customers, applications and products in the medium and long-term. The seeding of the market is enhanced when the standards are publicly available - they are easier to be taken up by non-members. In standardizing, companies potentially give up a competitive edge in the short-term goal but regain it in the medium to long term by producing products that add value and enhance the standard in a much larger market.

Standards need to be developed at the right time. The development of new products and markets seems to follow a double peak of activity with a dip in between. The optimum time to standardize is towards the end of the first (research) peak or in the dip between the peaks, before the second (development) peak occurs. Standardizing too early before the research is finished can lead to immature, bad standards whereas standardizing too late, once companies have made significant investments, can mean the standards are ignored. We regard MAS to be somewhere in the dip between the peaks at this stage.

3.1 Agent-Based Interoperability

The electronic service space for e-commerce current and future direction requires advancement in distributed infrastructure. Key to this is the interoperability of intelligent distributed systems. This paper provides an initial analysis of the requirements for high-level interoperability of services in an intelligent yet distributed and dynamic manner.

Interoperability is at many levels and this approach represents interoperability as a high-level. The importance of this is to recognize all levels and the impact of each level on reaching semantic integration, which is both flexible and extendable. Often, the focus on just one level (syntax) and which limits the possibilities of providing the type of infrastructure necessary for dynamic and new e-services.

In particular we wanted to bring out the following important features:

- General components can be designed to map to interoperability needs,
- Openness, which can be achieved through providing structured and explicit interfaces of interoperability i.e. a standard is essential. Although business models may not want complete openness for economic reasons it is necessary to have a certain degree to support services on the advancing internet in any meaningful way
- Frequently, interoperability in agent systems is only considered at some communication level. This is not enough for the future of service support but is an essential starting point.
- The meaning of an action of communicating is at a different level of interoperability – it is more than just sending a message. The content of the message requires an ontological definition. However, a ontological definition can vary to such an extent that semantic interoperability can not be achieved and certainly service integration in any open sense is lost. Ontologies are still defined and abstracted by hand. The automation of this process is still a main issue but aspects are being resolved – see [1] and [15]

Although standards (for specifications) can underpin interoperability, well-defined specifications in themselves are not sufficient for interoperability - general characteristics of the specifications such as scope, extensibility and form will impact the interoperability and these need to be assessed. As the relationship between the form of the specification and interoperability has already been addressed in Section 2, the scope and extensibility aspects will be debated in the remainder of this section. A discussion of the content of the specifications will be deferred until Section 4.

3.2 The Scope of Standards

The content and boundary of agent specifications is contentious for several reasons: because the type and variety of agents supported and the type of interfaces to the supporting computation and communication infrastructure for agents, are debatable. Here an overview of the scope and content of the FIPA agent specifications will be debated. First some general statements are made then some of detail of FIPA's approach will be explained in each case.

FIPA focuses on specifying external communication between agents rather than the (internal) processing of the communication at the receiver. The organisational schematic model in Figure 1 illustrates this: FIPA agent specifications cover four main areas: agent communication, core support for agent communication, other support for communication and application domains. FIPA has not standardised the internal processing of communication because it is too contentious, at least at this time. instance, it would be difficult to define a sufficiently generalised, and hence standardised, inference engine that could infer what to do when particular types of agent messages are received.

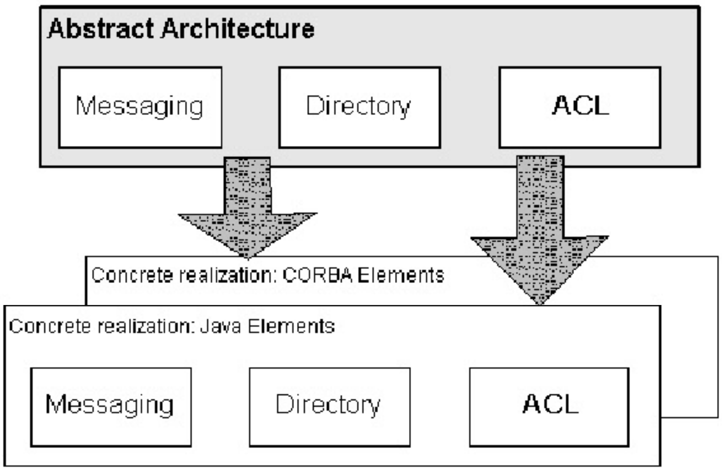

Abstract Architecture
Messaging
Directory
ACL
Concrete realization: CORBA Elements
Concrete realization: Java Elements
Messaging
Directory
ACL

Fig. 3. The abstract architecture and its realisation

This introduces a second important point that *FIPA specifications, foremost, attempt to cover generalisations and high-level neutral abstractions*. The core FIPA specifications are neutral with respect to:

- a specific service or end-user application;
- a particular supporting software and hardware infrastructure and implementation such as a computer language.

For example the FIPA Abstract Architecture [6] defines a high-level organisational model for agent communication and core support (minimum support required) for communication such as a directory service, message transport service and namespace. The abstract architecture is neutral with respect to any particular directory service or the use of a particular network protocol for message transport. The abstract architecture itself cannot be directly implemented, but instead forms the basis for the development of concrete architectural specifications (see Figure 3). Concrete implementations can implement all or part of the specification.

Thirdly, specifications are often dependent on other horizontal layers, e.g. they may use an existing software infrastructure. *A key issue is how much leverage to make from existing (non-agent) technology and how to link the agent parts to an existing infrastructure*. The scope of MA specifications generally includes the interpretation and handling of ACL (Agent Communication Language) messages, facilitator agents, and the use of, but not the actual specification of an existing software infrastructure such as message transport protocols, and message persistence schema, to underpin agent communication.

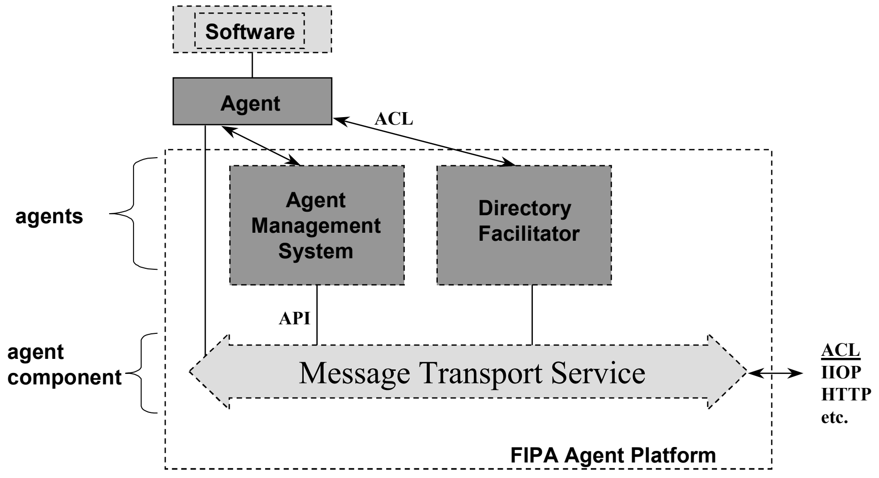


Fig. 4. The FIPA agent platform

Fourthly, different levels of granularities need to be modelled: we need to distinguish between agents and the level of granularity below that - the components that form the agent but, which in isolation are not an agent. *A related issue is determining what features should be modelled as agents and which should be modelled as non-agent parts.*

The following is an example of something that was first modelled as an agent but was later modelled as a non-agent part. The initial FIPA agent architecture was in essence taken to correspond to the agent platform defined in the agent management specification [12]. In the early versions of the specification (1997-1998), the message transport specification was modelled as an agent. At first sight this seemed to offer great flexibility: because the transport agent is an agent, all agents can interact with it in a standard way and with a potentially very flexible and semantically rich ACL messages. But there is a downside – efficiency. To transfer a single message between agents always required sending at least two messages, one to ask the agent transport agent to send a message, the other for the agent transport to actually send the message. Hence, in later versions of the Agent Management Specification [12], the message transport is a non-agent service that can be invoked via some Application Programmer's Interface rather than via an ACL message (agent interface), see Figure 4.

It is also worth mentioning that the FIPA Abstract Architecture [6] requires no single generic service such as naming and the directory service to be an agent (but they may be an agent), whereas the FIPA agent management specification mandates the use of a combined name and agent life-cycle management service (called the Agent Management Service or AMS) and the directory service (called the Directory Facilitator or DF) to be agents. Although, these appear to be mutually exclusive, they are not necessarily so. If agent management is present (it is optional according to the Abstract Architecture Specification) and it adheres to the agent management specification, then it requires agent based name and agent based directory services. If both the name service and directory service (and transport service) adhere to the properties and interfaces defined in the Abstract Architecture then the particular use of the agent management specification can also adhere to the FIPA Abstract Architecture specification.

3.3 Extensibility, Openness, Interoperability and Boot-Strapping

It is highly desirable that agent specifications are sufficiently extensible and open to work in a heterogeneous and changing world. This extensibility and openness is desirable at two distinct levels of granularity: at the agent level and at the agent component level (the components that underpin the agent).

At the agent-level, if agent systems are to scale up in the market-place, leading to mass-market penetration, then openness with respect to multiple vendors being free to add new agents and aggregate agents within a market-place, collaboratively, competitively and dynamically is highly attractive [17,18].

At the agent component level, it is desirable that the interface between the agent component and the agent does not bind the agent to a single particular instance of the agent component. For example, let's consider the agent transport as an agent component. In early versions of the Agent Transport Specification (in 1997 to 1998, it was actually part of the Agent Management Specification), FIPA specified the use of a single so called base-line message transport, the Object Management Group IIOP transport, which was ideal for use in low volume transaction, wire-line, private networks (without firewalls). However when FIPA agents were being considered for use via firewalls, for high-transaction processing and for wireless environments, the IIOP transport was considered to be less than ideal. It then became clear that agent component interfaces needed to neutral and abstract, e.g., the Agent transport specification (and the Abstract Architecture specification) can support multiple message transport protocols.

This openness to support bind agents to multiple instances of non-agent components such as multiple agent transport protocols introduces new interoperability problems. How can we bootstrap agent systems that are potentially using multiple transports? The Abstract architecture says that gateways will be needed to interlink different types of non-agent component but it does not deal in detail with bootstrapping. The specification of a base-line protocol, or in general the specification of a mandatory type of non-agent component that must always be there is useful because it means for the case of the transport protocol, the IIOP base-line could be used to negotiate the use of a more optimal transport protocol during the session. However, the use of a baseline even for bootstrapping may be unsuitable.

4 FIPA Standards: Engineering Issues

4.1 Overview of the Specifications

We have classified and packaged the agent specifications into four groups (Figure 1):
1. FIPA agent communication language (ACL) specifications
2. FIPA core communication support services

3. Other FIPA Communication support services
4. FIPA Application services.

The two sets of specifications that fundamentally define FIPA agent systems are groups 1 and 2 (2 corresponds to the FIPA Abstract Architecture [6]). In practice all of the currently open source implementations also adhere to two specifications in group 3: the Agent Management Specification [12] and the FIPA Message Transport Specification [7].

The specifications in groups 1 and 3 separate an abstract interface from a particular concrete realisation to promote specification extension and maintenance. For example, if another content language or message transport is added, the abstract interface specification part remains unchanged and a new concrete part is added.

Groups 1 and 3 are considered in more detail below. The second group has already been partially discussed in the scope section earlier. The fourth group of application services (see also Figure 4) provides an excellent starting point to understanding how the FIPA approach works in practice by using a single worked example for illustration within a single (application) specification document.

4.2 FIPA Agent Communication Language Specifications

The FIPA agent communication specifications also referred to as the the *Agent Communication Language* (ACL), are based on speech act theory [18] consist of:

- a fixed core set of about speech acts or communicate act messages
- a fixed core set of interaction protocols (the FIPA specifications refer to these sometimes as just "protocols")
- an extensible set of content languages.

N.B. The use of the term Agent Communication Language is overloaded: sometimes it is used to refer just to the speech acts, sometimes to the combination of the speech acts, interaction protocols and sometimes to the encoding of this combination for message transfer.

The set of speech acts form the basic set of message types exchanged between agents. Often messages are more usefully modelled within the context of a dialogue, an exchange of several messages between the sender and receiver. For example, client-server systems frequently exchange messages within a request-reply dialogue. The FIPA interaction protocols support richer dialogues and include task allocation (contract net), negotiation (auctions) and active directories (subscription).

The content is the part of a message that represents the domain dependent component of the communication. A content language is used to encode the *content* of a message. FIPA does not provide a single mandatory content language, but a set of reference content languages for a heterogeneous world such as Semantic Language, Choice Constraint Language (CCL), Knowledge Interchange Format (KIF) and W3C's Resource Description Framework (RDF). It is implied that the terms used in the content language are defined in an explicit ontology (at a simple level, the ontology can be considered as a domain specific dictionary of terms and the definition of any relationships between those terms) referenced in the ACL.

Finally, two levels of communicative syntactical wrapper are used when transmitting the message in practice, the first of these is sometimes (also) referred to as the ACL message structure [19], this defines the sender, receiver, the communicative act, interaction protocol, ontology, content language and the content for the current message. A second or outer syntactical wrapper is used to wrap this message for message transport.

4.3 FIPA Specification Development and Maintenance

Specification and standard development requires a structured approach. A more structured process and life-cycle for specification development and maintenance was introduced in 2000 in part to address the following issues, pre-2000:

- it was not clear prior to 2000 how mature a specification is: whether or not it had just been released, whether it was currently being evaluated or whether or not the specifications had been road-tested;
- the design decisions for modifying the specifications were not captured and indexed.

FIPA specifications are given a status which defines their position in FIPA's specification life cycle (Figure 5). A specification begins life as a Preliminary status specification by a Technical Committee when it is working on a draft. When this TC decides that the draft is ready for implementation, it may be advanced to Experimental status and is frozen for a period of two years. After successful implementations and field trials of this specification, it may be advanced to Standard status, which is considered to be a stable and mature specification ratified by FIPA and implemented by organisations and companies. If a specification becomes unnecessary or is superseded by another specifications, then it is Deprecated for a period to six months to allow developers to change their implementations accordingly. After this period, the specification is made Obsolete - this means that it is no longer a supported specification of FIPA.

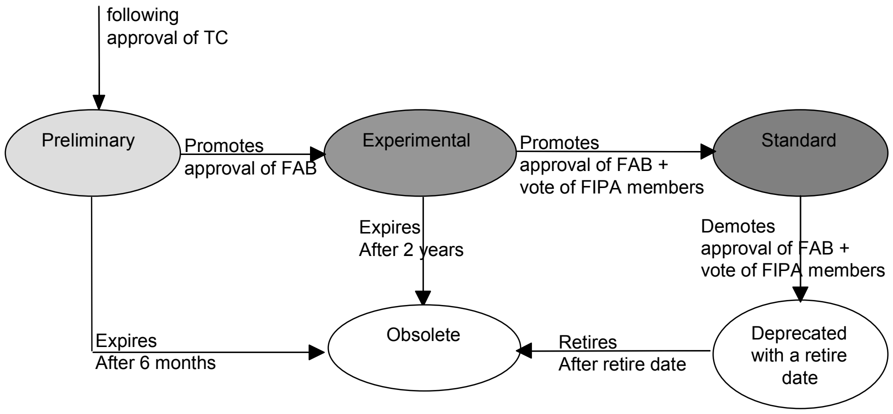


Fig. 5. Life-cycle of FIPA specifications. Specifications only attain standard status later in the life-cycle, following approval of the FIPA Architectural Board and the membership. TC indicates a Technical Committee.

FIPA believes in trying out the specifications in the field (experimental status) before they attain standard status. To the best of authors' knowledge, no specification has yet reached standard status, the most mature ones are only currently only experimental. This gives FIPA the option to investigate several potentially overlapping designs and then to adopt the most successful practical one.

FIPA specifications are developed and maintained on a continual basis. They are primarily discussed, reported and assigned a status (see below) at quarterly FIPA forum meetings. See [1] for details of the previous and future meetings. Non-member companies and organisations can attend FIPA meetings when contributing to FIPA's business.

5 Developing, Deploying and Implementing FIPA Specifications

FIPA specifications and the open source implementations have developed considerably to bring agent technology to world-wide deployment, enabling and promoting such projects as Agentcities. However, the FIPA specifications are not intended to be a complete blueprint or specification for building multi-agent system. For example, FIPA standards do not prescribe how to describe existential aspects of how agents in a discrete world, nor do they define error handling although some aspects of error reporting are covered. Some of the practical issues in using the FIPA standards are reported in [20]. Note that currently, there is no formal or clear mechanism to determine the compliance of a FIPA architecture implementation other than testing the interoperability of heterogeneous FIPA platforms in the field.

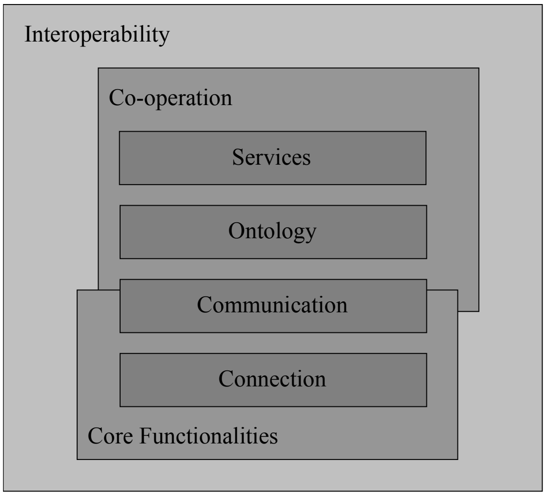


Fig. 6. Identification of interoperability layers [17]

Figure 6 summarises the interoperability layers that can be identified. From an interoperability perspective there are two key separations: core functionalities and cooperation. The core-functionalities for the design and development of agent and distributed systems are well understood and are dealt with by a number of Open Source platforms. A particular event that has harden these interoperability checks was at the London FIPA meeting in 2001 [1] , were a parallel session of testing between three platforms occurred: JADE [21], FIPA-OS [22] and ZEUS [23]. The result was two-fold: a) Illustrated the advancement of the FIPA specs as interoperability was more efficient and was able to test more of the specifications than the previous event in the 1999 meeting and b) FIPA specifications were considered fairly thorough, in that, only minor changes were necessary to enable the “core functions” to be tested.

To move towards the more application level of smart services requires the co-operative aspects to be addressed and this is known to be complex. Some progress has been made in this area, examples of this are seen in the specification of the DF, service ontology structure and protocols for service registrations in the FIPA specification [12] and hence the systems can perform conformance testing for interoperability at this level.

5.1 Concrete Architectures

There is inherent high complexity, cost and risk in developing a good practical standard and in developing systems that adhere to the standard. These act as a barrier to users and developers who wish to assess this new technology. Clearly, reference implementations are useful here and there seems to be (at least) two main approaches to develop these. Essentially, reference implementations can be developed under an open license or a closed license. We focus on the former approach.

An open source licence implementation can reduce the barrier for the adoption of FIPA standards, enhancing the ability of agent application developers to construct applications using FIPA technology. Minimizing cost is particularly important to encourage take up in education establishments and small medium enterprises. It can also improve software quality through third party development - more eyes, less bugs [24]. Users can directly reduce the time-scale for bug correction by providing their own fixes- this effect can be dramatic within a large user base. Open source is a powerful mechanism for engaging users while the market for FIPA agent technology is developing – it can be regarded as a form of Rapid Application Development in which new user requirements and actual extensions can be incremented during the development process. The April Agent Platform [25], FIPA-OS, JADE, and ZEUS are FIPA based agent systems released under an open source license and provides rich and flexible support for agent communication. These open source projects can be regarded as the first concrete realisations of the FIPA specifications.

Two additional initiatives to use the specifications are also worth mentioning Firstly, the JAS [26] project is developing a concrete realisation of the FIPA Abstract Architecture – this is work in progress at this time. Secondly, there are number of collaborative projects that are using, applying and evaluating the FIPA specifications – a selection of these are discussed in the next section.

5.2 Collaborative FIPA-Based Projects

There are a number of projects that have used the FIPA specifications and open source platforms for development, we summarise only a few of these activities here (see the FIPA web-site for more examples):

- The goal of the FACTS [27] project was to validate the work of FIPA and other standards groups by constructing a number of demonstrator systems based on FIPA's proposed standards. The focus of the project was on the interaction between different implementations of agents and agent platforms. The project was structured around two development cycles: during phase 1, agent interoperability was tested primarily within each of three application areas (audio-visual broadcasting and entertainment, service reservation, electronic commerce); during phase 2, agent interoperability was tested between the different application areas.
- LEAP [28] is the precursor of the second generation of FIPA agent platforms. Project LEAP (Lightweight Extensible Agent Platform) addresses the need for open infrastructures and services that support dynamic, mobile enterprises. It will develop innovative agent-based services supporting three requirements of a mobile enterprise workforce: Knowledge management (anticipating individual knowledge requirements), decentralised work co-ordination (empowering individuals, co-ordinating and trading jobs), travel management (planning and co-ordinating individual travel needs). It represents a major technical challenge – and has become the first integrated agent development environment capable of generating agent applications and executing them on a wide variety of run-time environments (implemented on devices such as computers, PDA and mobile 'phones) and communication mechanisms (TCP/IP, WAP etc.).
- The CRUMPET [29], Creation of User-friendly Mobile services PErsonalised for Tourism, Project takes advantage of integrating four key emerging technology domains and applying to the tourism domain: personalised services, multi-agent technology, location-aware services and transparent mobile data communication. CRUMPET is also a second generation FIPA agent platform based on FIPA-OS and adapted to small foot-print devices such as PDAs and to communication over wireless links.
- The Agentcities project [15] is to help realise the research and commercial potential of agent applications and accelerate the deployment of next generation Internet services by: (a) Deploying an infrastructure: building a standardised, publicly accessible, continually available, pan-European network of agent platforms providing the necessary connectivity and infrastructure to host agent-based services, (b) Deploying services: populating this network with a rich assortment of commercial grade agent-based services to form building blocks for advanced agent applications, and (c) Fostering research collaboration: promoting the network as a Europe wide focal point for agent research and development, thereby enabling cross-fertilisation between existing and future projects.

The sample projects outlined above each bring a new demand on the interoperability requirements of the FIPA specifications and delve further into the semantics of the co-operation interoperability aspects. It is very clear that in order to get a project such as Agentcities to work means that the semantics of ontologies, service interaction, commitments, trust etc. will have to be defined both as a specification and a design.

6 Conclusion and Further Work

The primary interoperability trials that have been performed between FIPA platforms within different projects and FIPA itself has pointed out differences between the implementations at all layers. A drawback of FIPA standards is that there is currently no means to certify a FIPA compliant platform. The main reasons for this certainly comes from the difficulty to validate platform behaviour through an interface that only allows messages to be exchanged. Another primary difficulty to perform validation tests is the need of a reference platform, which has to be defined by the standard bodies.

However, it is worth noting that FIPA has made major progress in advancing interoperability and addressing the issue of conformance testing. This concern of dealing with interoperability indirectly addresses some aspects of classical conformance testing, that is, at least the agent systems engaged in the interoperability tests conform in the same manner with FIPA specifications. The problem is that classical conformance testing in systems that exhibits the properties of autonomy and responsive behaviours, brings computational solutions to handling dynamic behaviour that cannot easily follow the classic model of conformance testing. In the dynamic and autonomous requirements of agent systems, we cannot rely upon "looking inside" of the agent system and analysing its computational behaviour. A company, business or end user will not want their communication strategy to be revealed. If we wish to trade the verifiability of an agent for autonomous behaviour then we need to consider other ways to test verification and conformance.

The key to moving in this direction of new modes of verification and conformance to allow the agent systems to be realistically applied to e-commerce's world is through considering social and organisational models. FIPA starts to address these organisational concerns through the domains and policies of models within the abstract architecture refinement. However, behaviour interpretation of these autonomous entities within a highly distributed environment requires a "social behaviour model". Part of the realisation of this is through the communication language and the semantics of commitments that FIPA starts to address within the technical committee for agreements. This process will expand the agent communication language that is currently specified within FIPA. However, this is not the whole story for realisation of right and wrong behaviours within the dynamic autonomous environment that is required for advancing the e-commerce's world. There are two other key features: models of trust and knowledge sharing.

Often in systems, the focus on concepts of trust is at the hard security level, and concerns theories of encryption and identification. However, there is a major difference between knowing that the data has been delivered without any interference and knowing that the entity that supplies that current data is reliable. The second issue of knowledge sharing sits under the hat of ontologies. Currently FIPA does not exploit some of the major advantages from having an ontology explicitly modelled and available. It still uses a reference model. Also, knowledge sharing and knowledge interoperability within agent systems has never yet been attained beyond any simple level. The next phase of FIPA will start to consider what features can be specified to advance the semantics of "social behaviour" and its convergence with domains, policies and commitments.

Acknowledgements

We wish to thank all the FIPA membership and collaborators, past and present, as the development and maintenance of the FIPA agent specifications discussed in this article is a joint effort. The opinions expressed in this article are those of the authors and do not necessarily reflect those of the FIPA membership at large. Our special thanks go to Stefan Channing at Queen Mary whose valuable comments helped to improve this paper.

References

1. FIPA, The Foundation for Intelligent Physical Agents, home web-page, http://www.fipa.org
2. Franklin, S., Graesser A.: "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents". In: Proceedings of Agent Theories, Architectures, and Languages, ECAI workshop, Budapest, Hungary, August, (1996) 193 - 206
3. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. In: Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3 No. 3 (2000) 285-312
4. Nwana H.S.: Software agents: an overview. Knowledge Engineering Review, Vol. 11, No.3, (1995) 205-244
5. Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. Knowledge Engineering Review, Vol. 10, No.2, (1995) 115-152
6. FIPA Abstract Architecture Specification. FIPA00001. FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents (2000)
7. FIPA Agent Message Transport Service Specification. FIPA00067. Foundation for Intelligent Physical Agents (2000)
8. FIPA Request Interaction Protocol Specification. FIPA00026. Foundation for Intelligent Physical Agents (2000)
9. Bauer, B., Mueller, J P. and Odell J: Agent UML: A formalism for specifying multiagent software systems. In: Ciancarini, P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering — Proceedings of the First International Workshop (AOSE-2000). Springer-Verlag, Berlin, Germany (2000)
10. Pitt, J., Kamara, L., Artikis A: Interaction Patterns and Observable Commitments in a Multi-Agent Trading Scenario. 5th Int. Conf on Autonomous Agents, Montreal, Canada (2001)
12. FIPA Agent Management Specification. FIPA00023. Foundation for Intelligent Physical Agents (2000)
13. Wooldridge, M.: Semantic Issues in the Verification of Agent Communication Languages. In: Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3, No. 1 (2000) 9-31
14. Singh, M.: Agent Communication Languages: rethinking the principles. IEEE Computer. Vol. 13, No. 12, (1998) 40-47
15. Agentcities. Testbed for a Worldwide Agent Network: EU Research and Development Project, IST-2000-28385, http://www.agentcities.org/ (2001)

16. Poslad, S., Buckle, P., Hadingham, R.G.: Open Source, Standards and Scaleable Agencies. Autonomous Agents 2000 Workshop on Infrastructure for Scalable Multi-agent Systems, Barcelona, June, (2000)
17. Charlton, P., Bonnefoy, D., Lhuilier, D.N.: Dealing with Interoperability for Agent-Based Services, Autonomous Agents, Montreal, Canada, (2001)
18. Searle, J. R.: Speech Acts. Cambridge University Press Cambridge, UK (1969)
19. FIPA ACL Message Structure Specification. FIPA00061. Foundation for Intelligent Physical Agents (2000)
20. Charlton, P., Cattoni, R .: Evaluating the Deployment of FIPA Standards when developing Application Services, IJPRAI, Vol. 15, No. 3, (2001)
21. Bellifemine, F., Rimassa, G., Poggi, A.: JADE - A FIPA-compliant Agent Framework. In: Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, London (1999)
22. Poslad, S. J., Buckle P., Hadingham R.: The FIPA-OS agent platform: Open Source for Open Standards. Proceedings of PAAM 2000, Manchester, UK, (2000) 355-368
23. Nwana, H., Ndumu, D., Lee, L., Collis, J.: ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems. In Applied Artifical Intelligence Journal, Vol. 13, No. 1, (1999) 129-186
24. Cathedral and Bazaar. HYPERLINKhttp://www.tuxedo.org/~esr/writings/cathedral-bazaar/
25. April Agent Platform. HYPERLINKhttp://sourceforge.net/projects/networkagent/
26. Java Agent Services, JSR-000087, HYPERLINKhttp://www.java-agent.org.
27. FACTS: FIPA Agent Communication Technologies and Services, EU Project Number ACTS-1997- AC317. http://www.labs.bt.com/projects/facts/
28. LEAP: “Lightweight Extensible Agent Platform”, EU Project Number IST-1999-10211, http://www.cordis.lu/ist/projects/99-10211.htm.
29. Poslad, S., Laamanen, H., Malaka, R., Nick, A., Buckle, P., Zipf, A.: CRUMPET: Creation of User-friendly Mobile services PErsonalised for Tourism. 3G2001 IEE conference, London, (2001)

Distributed Problem Solving and Planning

Edmund H. Durfee[1]

Artificial Intelligence Laboratory
EECS Department
University of Michigan
Ann Arbor, MI 48104, USA
durfee@umich.edu

Abstract. Distributed problem solving involves the collective effort of multiple problems solvers to combine their knowledge, information, and capabilities so as to develop solutions to problems that each could not have solved as well (if at all) alone. The challenge in distributed problem solving is thus in marshalling the distributed capabilities in the right ways so that the problem solving activities of each agent complement the activities of the others, so as to lead efficiently to effective solutions. Thus, while working together leads to distributed problem solving, there is also the distributed problem of how to work together that must be solved. We consider that problem to be a distributed planning problem, where each agent must formulate plans for what it will do that take into account (sufficiently well) the plans of other agents. In this paper, we characterize the variations of distributed problem solving and distributed planning, and summarize some of the basic techniques that have been developed to date.

1 Introduction

Distributed problem solving is the name applied to a subfield of distributed AI in which the emphasis is on getting agents to work together well to solve problems that require collective effort. Due to an inherent distribution of resources such as knowledge, capability, information, and expertise among the agents, an agent in a distributed problem-solving system is unable to accomplish its own tasks alone, or at least can accomplish its tasks better (more quickly, completely, precisely, or certainly) when working with others.

Solving distributed problems well demands both group **coherence** (that is, agents need to have the incentive to work together faithfully) and **competence** (that is, agents need to know how to work together well). Group coherence is hard to realize among individually-motivated agents, but without it it is difficult to view the multiple agents as solving a problem in a distributed way. In distributed problem solving, therefore, we typically assume a fair degree of coherence is already present: the agents have been designed to work together; or the payoffs to self-interested agents are only ac-

[1] This is an abridged and updated version of [15], which is Chapter 3 in Weiss' MultiAgent Systems: A Modern Approach to Distributed Artificial Intelligence, published by MIT Press in 1999 [52].

crued through collective efforts; or social engineering has introduced disincentives for agent individualism, etc. Distributed problem solving thus concentrates on competence; as anyone who has played on a team, worked on a group project, or performed in an orchestra can tell you, simply having the desire to work together by no means ensures a competent collective outcome!

Distributed problem solving presumes the existence of problems that need to be solved and expectations about what constitute solutions. For example, a problem to solve might be for a team of (computational) agents to design an artifact (say, a car). The solution they formulate must satisfy overall requirements (it should have four wheels, the engine should fit within the engine compartment and be powerful enough to move the car, etc.), and must exist in a particular form (a specification document for the assembly plant). The teamed agents formulate solutions by each tackling (one or more) subproblems and synthesizing these subproblem solutions into overall solutions.

Sometimes the problem the agents are solving is to construct a plan. And often, even if the agents are solving other kinds of problems, they also have to solve planning problems as well. That is, how the agents should plan to work together---decompose problems into subproblems, allocate these subproblems, exchange subproblem solutions, and synthesize overall solutions---is itself a problem the agents need to solve. Distributed planning is thus tightly intertwined with distributed problem solving, being both a problem in itself and a means to solving a problem.

In this paper, we describe the concepts and algorithms that comprise the foundations of distributed problem solving and planning. We outline how protocols protocols for interaction can be used to ensure the right information is conveyed to the right agents to accomplish distributed problem solving and planning. We assume that the reader is familiar with traditional AI search techniques; since problem solving and planning are usually accomplished through search, we make liberal use of the relevant concepts.

The remainder of this paper is structured as follows. We begin by introducing some representative example problems, as well as overviewing a variety of other applications of the techniques to be described. Working from these motivating examples, we work our way up through a series of algorithms and concepts as we introduce increasingly complicated requirements into the kinds of problems to solve, including planning problems.

2 Example Problems

There are several motivations for distributed problem solving and distributed planning. One obvious motivation is that using distributed resources concurrently can allow a speedup of problem solving thanks to parallelism. The possible improvements due to parallelism depend, of course, on the degree of parallelism inherent in a problem.

One problem that permits a large amount of parallelism during planning is a classic toy problem from the AI literature: the **Tower of Hanoi** (ToH) problem (see Figure 1). As the reader will recall from an introductory AI course, ToH consists of 3 pegs and n disks of graduated sizes. The starting situation has all of the disks on one peg, largest at bottom to smallest at top. The goal is to move the disks from the start peg to another peg, moving only one disk at a time, without ever placing a larger disk on top of a smaller disk. The problem, then, is to find a sequence of moves that will achieve the goal state.

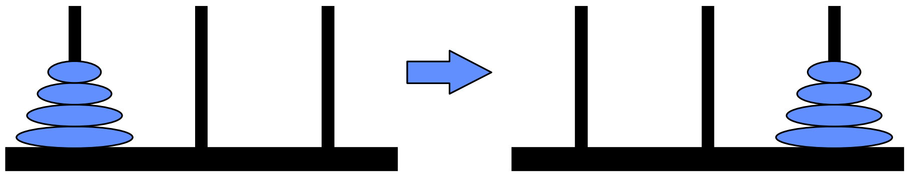


Fig. 1: Tower of Hanoi

A second motivation for distributed problem solving and planning is that expertise or other problem-solving capabilities can be inherently distributed. For example, in concurrent engineering, a problem could involve designing and manufacturing an artifact (such as a car) by allowing specialized agents to individually formulate components and processes, and combining these into a collective solution. Or, supervisory systems for air-traffic control, factory automation, or crisis management can involve an interplay between separate pieces for event monitoring, situation assessment, diagnosis, prioritization, and response generation. In these kinds of systems, the problem is to employ diverse capabilities to solve problems that are not only large (the ToH can itself be arbitrarily large) but also multi-faceted.

As a simple example of distributed capability, we will use the example of **distributed sensor network establishment** for monitoring a large area for vehicle movements. In this kind of problem, the overall task of monitoring cannot be done in a central location since the large area cannot be sensed from any single location. The establishment problem is thus to decompose the larger monitoring task into subtasks that can be allocated appropriately to geographically distributed agents.

A third motivation is related to the second, and that is that beliefs or other data can be distributed. For example, following the successful solution of the distributed sensor network *establishment* problem just described, the problem of actually doing the **distributed vehicle monitoring** could in principle be centralized: each of the distributed sensor agents could transmit raw data to a central site to be interpreted into a global view. This centralized strategy, however, could involve tremendous amounts of unnecessary communication compared to allowing the separate sensor agents to formulate local interpretations that could then be transmitted selectively.

Finally, a fourth motivation is that the results of problem solving or planning might need to be distributed to be acted on by multiple agents. For example, in a task involving the delivery of objects between locations (figure 2), **distributed delivery** agents can act in parallel. The formation of the plans that they execute could be done at a centralized site (a dispatcher) or could involve distributed problem-solving among them. Moreover, during the execution of their plans, features of the environment that were not known at planning time, or that unexpectedly change, can trigger changes in what the agents should do. Again, all such decisions could be routed through a central coordinator, but for a variety of reasons (exploiting parallelism, sporadic coordinator availability, slow communication channels, etc.) it could be preferable for the agents to modify their plans unilaterally or with limited communication among them.

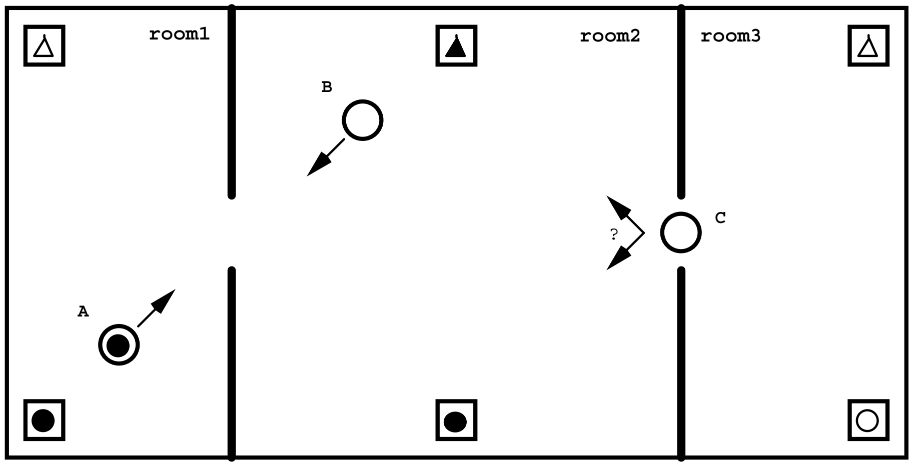


Figure 2: Distributed Delivery Example

In the above, we have identified several of the motivations for distributed problem solving and planning, and have enumerated examples of the kinds of applications for which these techniques make sense. In the rest of this paper, we will refer back to several of these kinds of application problems, specifically:
Tower of Hanoi (ToH)
Distributed Sensor Network Establishment (DSNE)
Distributed Vehicle Monitoring (DVM)
Distributed Delivery (DD)

3 Task Sharing

The first class of distributed problem-solving strategies that we will consider have been called “task sharing” or “task passing” strategies in the literature. The idea is simple. When an agent has many tasks to do, it should enlist the help of agents with few or no tasks. The main steps in task sharing are:

1. **Task decomposition**: Generate the set of tasks to potentially be passed to others. This could generally involve decomposing large tasks into subtasks that could be tackled by different agents.
2. **Task allocation**: Assign subtasks to appropriate agents.
3. **Task accomplishment**: The appropriate agents each accomplish their subtasks, which could include further decomposition and subsubtask assignment, recursively to the point that an agent can accomplish the task it is handed alone.

4. **Result synthesis**: When an agent accomplishes its subtask, it passes the result to the appropriate agent (usually the original agent, since it knows the decomposition decisions and thus is most likely to know how to compose the results into an overall solution).

Note that, depending on the circumstances, different steps might be more or less difficult. For example, sometimes an overburdened agent begins with a bundle of separate tasks, so decomposition is unnecessary; sometimes the agent can pass tasks off to any of a number of identical agents, so allocation is trivial; and sometimes accomplishing the tasks does not yield any results that need to be synthesized in any complex way.

3.1 Task Sharing Among Homogeneous Agents

To get a feel for the possibilities of task sharing, we start with the very simple ToH problem. Consider the task-sharing steps when it comes to this problem:

1. Task decomposition: Means-ends analysis, where moving the largest disk that is not at its destination peg is considered the most important difference, leads to a recursive decomposition: solve the problem of getting to the state where the largest disk can be moved, and get from the state after it is moved to the goal state. These subproblems can be further decomposed into problems of moving the second largest disk to the middle peg to get it out of the way, so the state where that can be done needs to be reached, etc.
2. Task allocation: If we assume an indefinite number of identical idle agents capable of solving (pieces of) the ToH problem, then allocation reduces to just assigning a task randomly to one of these idle agents. If the decomposed problems are such that the start and goal states are the same (that is, where the most significant difference is also the only difference), then the recursive decomposition terminates.
3. Task accomplishment: In general, an agent can use means-ends analysis to find the most significant difference between the start and goal states that it is responsible for, and will decompose the problem based on these.
4. Result synthesis: When an agent has solved its problem, it passes the solution back on up. When an agent has received solutions to all of the subproblems it passed down, it can compose these into a more comprehensive sequence of moves, and then pass this up as its solution.

ToH represents an ideal case of the possibilities of distributed problem solving due to the hierarchical nature of the problem. Unfortunately, most problems are more complicated than ToH for reasons including:

1. Problems will often require backtracking back upward in the abstraction hierarchy, because the solution of one can affect the solution of others.
2. The number of available (idle) agents is not indefinite, and will generally be fixed rather than scaling up with the problem size.

3. Problems usually cannot be decomposed into equal-sized subproblems, especially in domains where problems are decomposed into qualitatively different pieces requiring different expertise.
4. The processes of decomposing problems, distributing subproblems, and collecting results can require substantial time

3.2 Task Sharing in Heterogeneous Systems

One of the powerful motivations for distributed problem solving is that it is difficult to build artifacts (or train humans) to be competent in every possible task. Moreover, even if it feasible to build (or train) an omni-capable agent, it is often overkill because, at any given time, most of those capabilities will go to waste. The strategy in human systems, and adopted in many distributed problem-solving systems, is to bring together on demand combinations of specialists in different areas to combine their expertise to solve problems that are beyond their individual capabilities.

In the ToH example, the subproblems required identical capabilities, and so the decisions about where to send tasks was extremely simple. When agents can have different capabilities, and different subproblems require different capabilities, then the assignment of subproblems to agents is not so simple.

Conceptually, it is possible for an agent to have a table that identifies the capabilities agents, so that it can simply select an appropriate agent and send the subproblem off, but usually the decisions need to be based on more dynamic information. For example, if several candidate agents are capable of solving a subproblem, but some are already committed to other subproblems, how is this discovered? One way is to use the *Contract Net protocol*: An agent (in Contract-Net terms, the *manager*) decomposes its larger problem into a set of subproblems, and announces each subproblem to the network, along with specifications about which other agents are eligible to accept the subproblem and how they should specify a bid for the subproblem. A recipient of the announcement decides whether it is eligible, and if so it formulates a bid. The manager collects bids, and awards the subproblem to the agent(s) (called *contractors*) with the best bid(s). A contractor receives the subproblem details, solves the subproblem (perhaps by in turn breaking it into subproblems and contracting those out), and returns the solution to the manager.

In the kind of open environment for which Contract Net was envisioned, it is unlikely that a manager will be acquainted with all of the possible contractors in its world. Thus, broadcasting the task announcements makes the most sense. However, to minimize the communication in the network, it can improve performance if a manager can direct an announcement to a subset of potential contractors (or in the extreme case, a single preferred contractor) based on prior experience. Directed contracting can thus be thought of as caching results of previous broadcasts. Since the cached results can become outdated, many implementations of Contract Net do not include this function. It is interesting to note, however, that this kind of capabilities database has found renewed favor in knowledge sharing efforts such as KQML [38], where some agents explicitly adopt the task of keeping track of what other agents purport to be good at.

When no contractors are found through the Contract Net protocol, one very simple response is for the manager to retry the announcement periodically, assuming that a contractor is out there but is currently occupied. The retry interval then becomes an important parameter: if retries happen too slowly, then many inefficiencies can arise as agents do not utilize each other well; but if retries happen to quickly, the network can get bogged down with messages. One strategy for overcoming such a situation is to turn the protocol on its head. Rather than announcing tasks and collecting bids, which implies that usually there are several bidders for each task, instead the protocol can be used by potential contractors to announce availability, and managers can respond to the announcements by bidding their pending tasks! It is possible to have a system alternate between the task and availability announcement strategies depending on where the bottlenecks are in the system at various times [48].

An alternative to retrying, especially when there might not be eligible contractors on the network, is for the manager could engage in iterative revision of its announcement, relaxing eligibility requirements until it begins to receive bids. An interesting aspect of this relaxation process is that the eligibility specifications could well reflect preferences over different classes of contractors - or, more specifically, over the quality of services that different contractors provide. In concert with other methods of handling a lack of bids (described above), a manager will be deciding the relative importance of having a preferred contractor eventually pursue the subproblem compared to finding a suboptimal contractor sooner. In many cases, these preferences and tradeoffs between them can be captured using economic representations. By describing parts of its marginal utility curve, for example, a manager can provide tradeoff information to an auction, which can then apply principled algorithms to optimize the allocation of capabilities, such as market mechanisms [53].

Finally, the difficulty a manager faces might be that no agent can do the kind of subproblem it is announcing. The manager could therefore instead try decomposing the overall problem differently such that contractors are available for the alternative subproblems. In general, the relationship between problem decomposition and subproblem allocation is extremely complex and has not received sufficient attention. Sometimes a manager should first determine the space of alternative contractors to focus problem decomposition, while other times the space of decompositions can be very restrictive. Moreover, decisions about the number of problems to decompose into and the granularity of those subproblems will depend on other features of the application environment, including communication delays. We say no more about these issues here, other than to stress the research opportunities in this area.

3.3 Task Sharing for DSNE

Smith and Davis (and others since) have explored the use of the Contract Net protocol for a variety of problems, including the Distributed Sensor Net Establishment (DSNE) problem [6]. To give the reader a flavor of this approach, we briefly summarize the stages of this application.

At the outset, it is assumed that a particular agent is given the task of monitoring a wide geographic area. This agent has expertise in how to perform the overall task, but is incapable of sensing all of the area from its own locality. Therefore, the first step is that an agent recognizes that it can perform its task better (or at all) if it enlists the help of other agents. Given this recognition, it then needs to create subtasks to offload to other agents. In the DSNE problem, it can use its representation of the structure of the task to identify that it needs sensing done (and sensed data returned) from remote areas. Given this decomposition, it then uses the protocol to match these sensing subtasks with available agents. It announces (either directed, focused, or broadcast) a subtask; we leave out the details of the message fields.

The important aspects of the announcement for our purposes here are the eligibility specification, the task abstraction, and the bid specification. To be eligible for this task requires that the bidding agent have a sensor position within the required sensing area identified and that it have the desired sensing capabilities. Agents that meet these requirements can then analyze the task abstraction (what, at an abstract level, is the task being asked of the bidders) and can determine the degree to which it is willing and able to perform the task, from its perspective. Based on this analysis, an eligible agent can bid on the task, where the content of a bid is dictated by the bid specification.

The agent with the task receives back zero or more bids. If it gets back no bids, then it faces the options previously described: it can give up, try again, broaden the eligibility requirements to increase the pool of potential bidders, or decompose the task differently to target a different pool of bidders. If it gets back bids, it could be that none are acceptable to it, and it is as if it got none back. If one or more is acceptable, then it can award the sensing subtask to one (or possible several) of the bidding agents. Note that, because the agent with the task has a choice over what it announces and what bids it accepts, and an eligible agent has a choice over whether it wants to bid and what content to put into its bid, no agent is forced to be part of a contract. The agents engage in a rudimentary form of negotiation, and form teams through *mutual selection*.

Using a distributed sensor network to perform vehicle monitoring, the runtime relationships between what is being monitored in different areas is as variable as the possible movements of vehicles through the areas. While a task-sharing strategy, exemplified in the Contract Net protocol, can *establish* a distributed sensor network, it does not provide a sufficient basis for using the network. Or, put more correctly, when task sharing is used to allocate *classes* of tasks among agents, then if different instances of those tasks have different interrelationships, discovering and exploiting those relationships requires the generation and sharing of tentative results.

4 Result Sharing

While a task-sharing strategy, exemplified in the Contract Net protocol, can *establish* a distributed sensor network, it does not provide a sufficient basis for using the network because the runtime relationships between what is being monitored in different areas is as variable as the possible movements of vehicles through the areas. Or, put more correctly, when task sharing is used to allocate *classes* of tasks among agents, then if different instances of those tasks have different interrelationships, discovering and exploiting those relationships requires the generation and sharing of tentative results.

By sharing results, problem solvers can improve group performance in combinations of the following ways:

1. **Confidence**: Independently derived results for the same task can be used to corroborate each other, yielding a collective result that has a higher confidence of being correct. For example, when studying for an exam, students might separately work out an exercise and then compare answers to increase confidence in their solutions.
2. **Completeness**: Each agent formulates results for whichever subtasks it can (or has been contracted to) accomplish, and these results altogether cover a more complete portion of the overall task. For example, in distributed vehicle monitoring, a more complete map of vehicle movements is possible when agents share their local maps.
3. **Precision**: To refine its own solution, an agent needs to know more about the solutions that others have formulated. For example, in a concurrent engineering application, each agent might separately come up with specifications for part of an artifact, but by sharing these the specifications can be further honed to fit together more precisely.
4. **Timeliness**: Even if an agent could in principle solve a large task alone, solving subtasks in parallel can yield an overall solution faster.

Accruing the benefits of result sharing obviously means that agents need to share results. But making this work is harder than you might think! First of all, agents need to know what to do with shared results: how should an agent assimilate results shared from others in with its own results? Second, given that assimilation might be nontrivial, that communicating large volumes of results can be costly, and that managing many assimilated results incurs overhead, agents should attempt to be as selective as possible about what they exchange. In the remainder of this section, we look at these issues.

4.1 Functionally Accurate Cooperation

In task-passing applications like ToH, the separate problem-solving agents are completely accurate in their computations (they have all information and a complete specification for their subtasks) and operate independently. In contrast, agents doing Distributed Vehicle Monitoring (DVM) lack information about what is happening elsewhere that could impact their calculations. As a result, these agents need to cooperate to solve their subtasks, and might formulate tentative results along the way that turn out to be unnecessary. This style of collective problem solving has be termed functionally-accurate (it gets the answer eventually, but with possibly many false starts) and cooperative (it requires iterative exchange) [33].

Functionally-accurate cooperation has been used extensively in distributed problem solving for tasks such as interpretation and design, where agents only discover the details of how their subproblem results interrelate through tentative formulation and iterative exchange. For this method to work well, participating agents need to treat the partial results they have formulated and received as tentative, and therefore might have to entertain and contrast several competing partial hypotheses at once. A variety of agent architectures can support this need; in particular, blackboard architectures [33] have often been employed as semi-structured repositories for storing multiple competing hypotheses.

Exchanging tentative partial solutions can impact completeness, precision, and confidence. When agents can synthesize partial solutions into larger (possibly still partial) solutions, more of the overall problem is covered by the solution. When an agent uses a result from another to refine its own solutions, precision is increased. And when an agent combines confidence measures of two (corroborating or competing) partial solutions, the confidence it has in the solutions changes. In general, most distributed problem-solving systems assume similar representations of partial solutions (and their certainty measures) which makes combining them straightforward, although some researchers have considered challenges in crossing between representations, such as combining different uncertainty measurements [58].

In functionally accurate cooperation, the iterative exchange of partial results is expected to lead, eventually, to some agent having enough information to keep moving the overall problem solving forward. Given enough information exchange, therefore, the overall problem will be solved. Of course, without being tempered by some control decisions, this style of cooperative problem solving could incur dramatic amounts of communication overhead and wasted computation. For example, if agents share too many results, a phenomenon called **distraction** can arise: it turns out that they can begin to all gravitate toward doing the same problem-solving actions (synthesizing the same partial results into more complete solutions). That is, they all begin exploring the same part of the search space. For this reason, limiting communication is usually a good idea, as is giving agents some degree of skepticism in how they assimilate and react to information from others. We address these issues next.

4.2 Distributed Constraint Satisfaction

One view of the iterative exchange of tentative solutions is as a distributed constraint satisfaction problem (DCSP). Each agent is trying to find values (solutions) for its variables (local subproblems) such that constraints between variables are satisfied (the local subproblem solutions "fit" together into a consistent global solution). It is typically assumed that agents whose variables have constraints between them know about this relationship, and therefore will communicate about their values with each other.

A variety of techniques for distributed constraint satisfaction has been developed over the years, and a complete description of them all is beyond the scope of this paper. The main challenges in solving a distributed constraint satisfaction problem are ensuring termination and completeness. That is, in the most general sense, what agents in a DCSP are doing is passing around their current proposed assignments of values to their variables; when constraint violations between their variables are detected, then some of them need to make different assignments. There is thus a danger that agents will make assignments and reassignments such that they never finish. Because a variable assignment involved in a constraint could still be correct (it might have been that other variables involved in the constraint were wrong), an agent should consider revisiting an assignment. But in the worst case, the combinations of local assignment decisions could lead to cycling through repetitions of collective assignments.

Similarly, local decisions about assignments and reassignments, without global oversight, could mean that some combinations of collective assignments might never be tried. Thus the DCSP search might be incomplete.

The solutions to these problems generally involve imposing some amount of global structure on the distributed search. Ensuring completeness is possible if agents have consistent knowledge about which of them should try new values for variables while others hold their values constant. This permits a systematic search of the space of collective assignments, meaning both that the search is complete and that it is possible to detect when all combinations have been tried so that the search can terminate. At the extreme, however, this can lead to a fully sequentialized search among the agents, eliminating any potential benefits of parallelism among agent activities. Techniques have been developed for increasing parallelism through asynchronous activities on the parts of the agents, while still ensuring systematic search [56].

The efficacy of the search, however, also depends on decisions about which agents will try alternative values while others hold still. Just as single-agent constraint satisfaction can employ heuristics like most-constrained-variable-first, DCSP can do so as well. This is of course complicated, however, since knowledge about the degree to which a variable is constrained might be acquired dynamically, and so the guidelines about which agents should change and which should hold still can change at runtime. Ensuring that each agent knows about these changes, and that a complete search can still be carried out, requires careful construction of protocols [56].

4.3 Shared Repositories and Negotiated Search

One strategy for reducing message passing is to instead concentrate tentative partial results in a single, shared repository. The blackboard architecture, for example, allows cooperating knowledge sources to exchange results and build off of them by communicating through a common, structured blackboard. This strategy has been adopted in a variety of distributed problem-solving approaches, including those for design applications [30] [54]. In essence, using a shared repository can support search through alternative designs, where agents with different design criteria can revise and critique the alternatives. In many ways, this is a distributed constraint satisfaction problem, but it differs from traditional formulations in a few respects.

Two important differences are: agents are not assumed to know whose constraints might be affected by their design choices, and agents can relax constraints in a pinch. The first difference motivates the use of a shared repository, since agents would not know whom to notify of their decisions (as is assumed in typical DCSP formulations). The second difference motivates the need for heuristics to control the distributed search, since at any given time agents might need to choose between improving some solutions, rejecting some solutions, or relaxing expectations (thus making some solutions that were previously considered as rejected now acceptable).

For example, agents engaged in negotiated search [30] have at their disposal a variety of operators for progressing the distributed problem-solving effort: *initiate-solution* (propose a new starting point for a solution); *extend-solution* (revise an already existing partial solution); *critique-solution* (provide feedback on the viability of an already existing partial solution); and *relax-solution-requirement* (change local requirements for solution acceptability). At any given time, an agent needs to decide which of these operators to apply, and where. While a systematic exploration of the space can be considered, the problem domains for negotiated search are typically complex enough that heuristic guidance is preferred. Heuristic measures for when to invoke operators (such as invoking the *relax-solution-requirement* operator when lack of progress is detected) and on what (such as relaxing requirements corresponding to the most constrained component) are generally application-specific.

4.4 Distributed Constrained Heuristic Search

Constraint satisfaction problems in distributed environments also arise due to contention for resources. Rather than assuming a shared repository for tentative partial solutions, a search strategy that has been gainfully employed for distributed resource allocation problems has been to associate an "agent" with each resource, and have that agent process the contending demands for the resource. One form that this strategy takes is so-called market-oriented programming [53] where associated with resources are auctions that support the search for equilibria in which resources are allocated efficiently. Market mechanisms are covered next.

A second form that this strategy takes is to allow resources to compute their aggregate demands, which then the competing agents can take into account as they attack their constraint-satisfaction problem. For example, distributed constrained heuristic search (DCHS) uses aggregate demand to inform a heuristic search for solving a distributed constraint satisfaction problem [50]. The idea is that more informed search decisions decrease wasted backtracking effort, and that constraint satisfaction heuristics such as variable and value ordering can be gainfully employed in a distributed environment.

DCHS works as follows:

1. An agent begins with a problem state comprised of a problem topology (the tasks to do and their relationships including constraints).
2. An agent propagates constraints within its state; it backtracks if an inconsistency is detected. Otherwise, it determines what resources it requires for what time intervals and computes a demand profile for those resources.
3. If the system is just beginning, or if the demand profiles differ from previous profiles, an agent sends the profile(s) to the resource(s).
4. A resource computes aggregate demand and informs the agents making the demands.
5. An agent uses the aggregate demands to order its variables (resource-and-time-interval pairs) and order the activities that it might assign to the highest-demand pair. It identifies a preferred resource/time-interval/activity assignment.
6. An agent requests that the resource reserve the interval for it.
7. The resource in turn grants the reservation if possible and updates the resource schedule. Otherwise the request is denied.

8. An agent processes the response from the resource. If the reservation is granted, the agent goes to step 2 (to propagate the effects of concretely scheduling the activity). If the reservation is not granted, the agent attempts another reservation, going to step 6.

This view of the search strategy, while simplified, highlights the use of resources being contended for to focus communication, and of an exchange of information that tends to decrease the amount of backtracking. That is, by giving agents an opportunity to settle the "difficult" contention issues first, much useless work is avoided in settling the easier issues and then discovering that these fail to allow the hard issues to be settled.

4.5 Market Mechanisms

There is a burgeoning literature on market-oriented mechanisms; here, we will just outline the basic ideas. As in the previous section, the underlying notion is that contention for resources can be captured as some measure of aggregate demand for the resource. In market mechanisms, this is more directly captured as the amount that each agent would be willing to pay for (a share of) the resource.

The market typically goes through multiple rounds of agents bidding on resources, and then the resources (or the auctions associated with them) providing information reflecting the aggregate demand. In the simplest terms, the auction announces a price for the resource, and the agents indicate how much they are willing to buy for that amount. As the system iterates, it converges (under some conditions) to an equilibrium where no agents change their resource demands given the current prices. At that point, an efficient allocation of resources is achieved.

Because it is founded on well-understood principles in economics, this kind of mechanism provides provable properties that are harder to establish in more heuristic methods. Of course, though, there are multiple challenges in making this kind of mechanism work for some of the more complicated problems for which heuristic methods have been developed. Extending the range of capabilities of market-based mechanisms is currently an extremely active research area, including dealing with combinatorial auctions (where the value agents place in items is dependent on getting combinations of items).

4.6 Organizational Structuring

When a shared repository cannot be supported or when problem-solving is not tantamount to resource scheduling, an alternative strategy for reducing communication is to exploit the task decomposition structure, to the extent that it is known. In a distributed design problem, for example, it makes sense to have designers working on components that must "connect" speak with each other more frequently than they speak with designers working on more remote parts of the design (of course, physical proximity might be only one heuristic!). Or, in a DVM task, agents monitoring neighboring parts of the space should communicate when their maps show activity at or near their mutual boundary. The notion is that agents have general roles to play in the collective effort, and by using knowledge of these roles the agents can make better interaction decisions.

This notion can be explicitly manifested in an organizational structure, which defines roles, responsibilities, and preferences for the agents within a cooperative society, and thus in turn defines control and communication patterns between them. From a global view, the organizational structure associates with each agent the types of tasks that it can do, and usually some prioritization over the types such that an agent that currently could do any of a number of tasks can identify the most important tasks as part of its organizational role. Allowing prioritization allows the structure to permit overlapping responsibilities (to increase the chances of success despite the loss of some of the agents) while still differentiating agents based on their primary roles.

Since each agent has responsibilities, it is important that an agent be informed of partial results that could influence how it carries out its responsibilities. More importantly, agents need not be told of results that could not affect their actions, and this can be determined based on the organizational structure. Thus, an organizational structure provides the basis for deciding who might potentially be interested in a partial result. It also can dictate the degree to which an agent should believe and act on (versus remain skeptical about) a received result.

While an organizational structure needs to be coherent from an overall perspective, it is important to note that, as in human organizations, an agent only needs to be aware of its local portion of the structure: what it is supposed to be doing (and how to decide what to do when it has choices), who to send what kinds of information to, who to accept what kinds of information from and how strongly to react to that information, etc. For practical purposes, therefore, organizational structures are usually implemented in terms of stored pattern-response rules: when a partial result that matches the pattern is generated/received, then the response actions are taken (to transmit the partial result to a particular agent, or to act on it locally, or to decrement its importance, etc.). Note that a single partial result could trigger multiple actions.

Finally, we have briefly mentioned that an organizational structure can be founded upon the problem decomposition structure, such as for the DSNE problem where agents would be made aware of which other agents are responsible for neighboring areas so that partial results that matched the overlapping regions of interest would be shared. The design of organizational structures for multi-agent systems, however, is generally a complex search problem in its own right. The search can be conducted in a bottom-up distributed manner, where boundaries between the roles of agents can be determined as the problem instance is initialized [7] or as problem solving progresses [26] [41], where adjustments to the structure can be based on reacting to performance inefficiencies of the current structure. In some cases, the organizational structure can be equated to a priority order for a distributed constraint satisfaction problem, and the agents are trying to discover an effective ordering to converge on a solution efficiently.

Alternatively, organizational structuring can be viewed as a top-down design problem, where the space of alternative designs can be selectively explored and candidate designs can be evaluated prior to their implementation [39] [5] [47]. The use of computational techniques to study, and prescribe, organizational structures is covered in [21].

4.7 Communication Strategies

Organization structures, or similar knowledge, can provide static guidelines about who is generally interested in what results. But this ignores timing issues. When deciding whether to send a result, an agent really wants to know whether the potential recipient is likely to be interested in the result now (or soon). Sending a result that is potentially useful but that turns out to not be at best clutters up the memory of the recipient, and at worst can distract the recipient away from the useful work that it otherwise would have done. On the other hand, refraining from sending a result for fear of these negative consequences can lead to delays in the pursuit of worthwhile results and even to the failure of the system to converge on reasonable solutions at all because some links in the solution chain were broken.

When cluttering memory is not terrible and when distracting garden paths are short, then the communication strategy can simply be to send all partial results. On the other hand, when it is likely that an exchange of a partial result will lead a subset of agents into redundant exploration of a part of the solution space, it is better to refrain, and only send a partial result when the agent that generated it has completed everything that it can do with it locally. For example, in a distributed theorem-proving problem, an agent might work forward through a number of resolutions toward the sentence to prove, and might transmit the final resolvent that it has formed when it could progress no further.

Between the extremes of sending everything and sending only locally complete results are a variety of gradations [10], including sending a small partial result early on (to potentially spur the recipient into pursuing useful related results earlier). For example, in the DVM problem, agents in neighboring regions need to agree when they map vehicles from one region to the other. Rather than waiting until it forms its own local map before telling its neighbor, an agent can send a preliminary piece of its map near the boundary early on, to stimulate its neighbor into forming a complementary map (or determining that no such map is possible and that the first agent is working down a worthless interpretation path).

So far, we have concentrated on how agents decide when and with whom to voluntarily share results. But the decision could clearly be reversed: agents could only send results when requested. Just like the choice between announcing tasks versus announcing availability in the Contract Net depends on which is more scarce, the same holds true in result sharing. When the space of possible interesting results is large compared to the actual results that are generated, then communicating results makes sense. But when the space of results formed is large and only few are really needed by others, then sending requests (or more generally, goals) to others makes more sense. This strategy has been explored in the DVM problem [5], as well as in distributed theorem proving [35] [20]. For example, in DARES [35], when a theorem proving agent would fail to make progress, it would request clauses from other such agents, where the set of desired literals would be heuristically chosen.

It is also important to consider the delays in iterative exchange compared to a blind inundation of information. A request followed by a reply incurs two communication delays, compared to the voluntary sharing of an unrequested result. But sharing too many unrequested results can introduce substantial overhead. Clearly, there is a tradeoff between reducing information exchanged by iterative messaging versus reducing delay in having the needed information reach its destination by sending many messages at the same time. Sen, for example, has looked at this in the context of distributed meeting scheduling [45]. Our experience as human meeting schedulers tells us that finding a meeting time could involve a series of proposals of specific times until one is acceptable, or it could involve having the participants send all of their available times at the outset. Most typically, however, practical considerations leave us somewhere between these extremes, sending several options at each iteration.

Finally, the communication strategies outlined have assumed that messages are assured of getting through. If messages get lost, then results (or requests for results) will not get through. But since agents do not necessarily expect messages from each other, a potential recipient will be unable to determine whether or not messages have been lost. One solution to this is to require that messages be acknowledged, and that an agent sending a message will periodically repeat the message (sometimes called "murmuring") until it gets an acknowledgment [18]. Or, a less obtrusive but more uncertain method is for the sending agent to predict how the message will affect the recipient, and to assume the message made it through when the predicted change of behavior is observed (see the later discussion of plan recognition).

4.8 Task Structures

Up to this point, we have made intuitive appeals to why agents might need to communicate results. The TAEMS work of Decker and Lesser has investigated this question much more concretely [8]. In their model, an agent's local problem solving can have non-local effects on the activity of other agents. Perhaps it is supplying a result that another agent must have to *enable* its problem-solving tasks. Or the result might *facilitate* the activities of the recipient, allowing it to generate better results and/or generate results faster. The opposites of these (*inhibit* and *hinder*, respectively) are among the other possible relationships.

By representing the problem decomposition structure explicitly, and capturing within it these kinds of task relationships, we can employ a variety of coordination mechanisms. For example, an agent that provides an enabling result to another can use the task structure representation to detect this relationship, and can then bias its processing to provide this result earlier. In fact, it can use models of task quality versus time curves to make commitments to the recipient as to when it will generate a result with sufficiently high quality. In situations where there are complex networks of non-local task interrelationships, decisions of this kind of course get more difficult. Ultimately, relatively static organizational structures, relationships, and communication strategies can only go so far. Going farther means that the problem-solving agents need to analyze their current situation and construct plans for how they should interact to solve their problems.

5 Distributed Planning

In many respects, distributed planning can be thought of simply as a specialization of distributed problem solving, where the problem being solved is to design a plan. But because of the particular features of planning problems, it is generally useful to consider techniques that are particularly suited to planning.

Distributed planning is something of an ambiguous term, because it is unclear exactly what is "distributed." It could be that the operative issue is that, as a consequence of planning, a plan is formulated that can be distributed among a variety of execution systems. Alternatively, the operative issue could be that the planning process should be distributed, whether or not the resulting plan(s) can be. Or perhaps both issues are of interest. In this section, we consider both distributed plans and distributed plan formation as options; we of course skip over the case where neither holds (since that is traditional centralized planning) and consider where one or both of these distributions exists.

5.1 Centralized Planning for Distributed Plans

Plans that are to be executed in a distributed fashion can nonetheless be formulated in a centralized manner. For example, a partial order planner can generate plans where there need not be a strict ordering between some actions, and in fact where those actions can be executed in parallel. A centralized coordinator agent with such a plan can break it into separate threads, possibly with some synchronization actions. These separate plan pieces can be passed (using task-passing technology) to agents that can execute them. If followed suitably, and under assumptions of correctness of knowledge and predictability of the world, the agents operating in parallel achieve a state of the world consistent with the goals of the plan.

Let us consider this process more algorithmically. It involves:

1. Given a goal description, a set of operators, and an initial state description, generate a partial order plan. When possible, bias the search to find a plan in which the steps have few ordering constraints among them.
2. Decompose the plan into subplans such that ordering relationships between steps tend to be concentrated within subplans and minimized across subplans. [31].
3. Insert synchronization (typically, communication) actions into subplans.
4. Allocate subplans to agents using task-passing mechanisms. If failure, return to previous steps (decompose differently, or generate a different partial order plan, ...). If success, insert remaining bindings into subplans (such as binding names of agents to send synchronization messages to).
5. Initiate plan execution, and optionally monitor progress (synthesize feedback from agents to ensure complete execution, for example).

Notice that this algorithm is just a specialization of the decompose-allocate-execute-synthesize algorithm used in task passing. The specific issues of decomposition and allocation that are involved in planning give it a special flavor. Essentially, the objective is to find, of all the possible plans that accomplish the goal, the plan that can be decomposed and distributed most effectively. But since the availability of agents for the subplans is not easy to determine without first having devised the subplans, it is not certain that the most decomposable and distributable plan can be allocated in any current context.

Moreover, the communication infrastructure can have a big impact on the degree to which plans should be decomposed and distributed. As an extreme, if the distributed plans require synchronization and if the communication channels are slow or undependable, then it might be better to form a more efficient centralized plan. The monetary and/or time costs of distributing and synchronizing plans should thus be taken into account. In practical terms, what this usually means is that there is some minimal subplan size smaller than which it does not make sense to decompose a plan. In loosely-coupled networks, this leads to systems with fewer agents each accomplishing larger tasks, while in tightly-connected (or even shared-memory) systems the degree of decomposition and parallelism can be increased.

5.2 Distributed Planning for Centralized Plans

Formulating a complex plan might require collaboration among a variety of cooperative planning specialists, just like generating the solution to any complex problem would. Thus, for complex planning in fields such as manufacturing and logistics, the process of planning could well be distributed among numerous agents, each of which contributes pieces to the plan, until an overarching plan is created.

Parallels to task-sharing and result-sharing problem solving are appropriate in this context. The overall problem-formulation task can be thought of as being decomposed and distributed among various planning specialists, each of which might then proceed to generate its portion of the plan. For some types of problems, the interactions among the planning specialists might be through the exchange of a partially-specified plan. For example, this model has been used in the manufacturing domain, where a general-purpose planner has been coupled with specialist planners for geometric reasoning and fixturing [28]. In this application, the geometric specialist considers the shape of a part to be machined, and generates an abstract plan as an ordering over the geometric features to put into the part. The general-purpose planner then uses these ordering constraints to plan machining operations, and the augmented plan is passed on to the fixture specialist, which ensures that the operations can be carried out in order (that the part can be held for each operation, given that as each operation is done the shape of the part can become increasingly irregular). If any of these planners cannot perform its planning subtask with the partially-constructed plan, they can backtrack and try other choices (see prior section on DCSPs). Similar techniques have been used for planning in domains such as mission planning for unmanned vehicles [13] and for logistics planning [55].

The more asynchronous activity on the part of planning problem-solvers that is characteristic of most distributed problem-solving systems can also be achieved through the use of result sharing. Rather than pass around a single plan that is elaborated and passed on (or discovered to be a deadend and passed back), a result-sharing approach would have each of the planning agents generate a partial plan in parallel and then share and merge these to converge on a complete plan in a negotiated search mode. For example, in the domain of communication networks, localized agents can tentatively allocate network connections to particular circuits and share these tentative allocations with neighbors [4]. When inconsistent allocations are noticed, some agents try other allocations, and the process continues until a consistent set of allocations have been found. In this example, result-sharing amounts to a distributed constraint satisfaction search, with the usual concerns of completeness and termination (see prior section on DCSPs).

5.3 Distributed Planning for Distributed Plans

The most challenging version of distributed planning is when both the planning process and its results are intended to be distributed. In this case, it might be unnecessary to ever have a multi-agent plan represented in its entirety anywhere in the system, and yet the distributed pieces of the plan should be compatible, which at a minimum means that the agents should not conflict with each other when executing the plans, and preferably should help each other achieve their plans when it would be rational to do so (e.g. when a helping agent is no worse off for its efforts).

The literature on this kind of distributed planning is relatively rich and varied. In this paper, we will hit a few of the many possible techniques that can be useful.

Plan Merging. We begin by considering the problem of having multiple agents formulate plans for themselves as individuals, and then having to ensure that their separate plans can be executed without conflict. Assume that the assignment of goals to agents has been done, either through task-sharing techniques, or because of the inherent distributivity of the application domain (such as in a distributed delivery (DD) task, where different agents are contacted by users to provide a delivery service). Now the challenge is to identify and resolve potential conflicts.

We begin by considering a centralized plan coordination approach. Let us say that an agent collects together these individual plans. It then has to analyze the plans to discover what sequences of actions might lead to conflicts, and to modify the plans to remove the conflicts. In general, the former problem amounts to a *reachability analysis*---given a set of possible initial states, and a set of action sequences that can be executed asynchronously, enumerate all possible states of the world that can be reached. Of these, then, find the subset of worlds to avoid, and insert constraints on the sequences to eliminate them.

In general, enumerating the reachable state space can be intractable, so strategies for keeping this search reasonable are needed. From the planning literature, many assumptions about the limited effects of actions and minimal interdependence between agents' goals can be used to reduce the search. We will look at one way of doing this, adapted from Georgeff [22] next.

As is traditional, assume that the agents know the possible initial states of the world, and each agent builds a totally-ordered plan using any planning technology. The plan is comprised of actions a_1 through a_n, such that a_1 is applicable to any of the initial states, and a_i is applicable in all states that could arise after action a_{i-1}. The state arising after a_n satisfies the agent's goal.

We represent an action as a STRIPS operator, with preconditions that must hold for the action to take place, effects that the action has (where features of the world not mentioned in the effects are assumed unaffected), and "during" conditions to indicate changes to the world that occur only during the action. The STRIPS assumption simplifies the analysis for interactions by allowing us to avoid having to search through all possible interleavings of actions; it is enough to identify specific actions that interact with other specific actions, since the effects of any sequence is just the combined effects of the sequence's actions.

The merging method thus proceeds as follows. Given the plans of several agents (where each is assume to be a correct individual plan), the method begins by analyzing for interactions between pairs of actions to be taken by different agents. Arbitrarily, let us say we are considering the actions a_i and b_j are the next to be executed by agents A and B, respectively, having arrived at this point through the asynchronous execution of plans by A and B. Actions a_i and b_j can be executed in parallel if the preconditions, during conditions, and effects of each are satisfiable at the same time as any of those conditions of the other action. If this is the case, then the actions can commute, and are essentially independent. If this is not the case, then it might still be possible for both actions to be taken but in a stricter order. If the situation before either action is taken, modified by the effects of a_i, can satisfy the preconditions of b_j, then a_i can precede b_j. It is also possible for b_j to precede a_i. If neither can precede the other, then the actions conflict.

From the interaction analysis, the set of unsafe situations can be identified. Clearly, it is unsafe to begin both a_i and b_j if they do not commute. It is also unsafe to begin a_i before b_j unless a_i has precedence over b_j. Finally, we can propagate these unsafe interactions to neighboring situations:

1. the situation of beginning a_i and b_j is unsafe if either of its successor situations is unsafe;
2. the situation of beginning a_i and ending b_j is unsafe if the situation of ending a_i and ending b_j is unsafe;
3. the situation of ending a_i and ending b_j is unsafe if both of its successor states are unsafe.

To keep this safety analysis tractable, actions that commute with all others can be dropped from consideration. Given a loosely-coupled multi-agent system, where agents mostly bring their own resources and capabilities to bear and thus have few opportunities to conflict, dropping commuting actions would reduce the agents' plans to relatively short sequences. From these simplified sequences, then, the process can find the space of unsafe interactions by considering the (exponential) number of interleavings. And, finally, given the discovered unsafe interactions, synchronization actions can be added to the plans to force some agents to suspend activities during regions of their plans that could conflict with others' ongoing actions, until those others release the waiting agents.

Plan synchronization need not be accomplished strictly through communication only. Using messages as signals allows agents to synchronize based on the completion of events rather than reaching specific time points. But many applications have temporal features for goals. Manufacturing systems might have deadlines for fabricating an artifact, or delivery systems might have deadlines for dropping off objects. For these kinds of applications, where temporal predictions for individual tasks are fundamentally important, the formulation of distributed plans can be based on scheduling activities during fixed time intervals. Thus, in these kinds of systems, the individual planners can formulate a desired schedule of activities assuming independence, and then plan coordination requires that the agents search for revisions to their schedules to find non-conflicting times for their activities (which can be accomplished by DCHS (see 3.0)). More importantly, different tasks that the agents pursue might be related in a precedence ordering (e.g. a particular article needs to be dropped off before another one can be picked up). Satisfying these constraints, along with deadlines and resource limitation constraints, turns the search for a workable collective schedule into a distributed constraint satisfaction problem.

A host of approaches to dealing with more complex forms of this problem exist, but are beyond the scope of this paper. We give the flavor of a few of these to illustrate some of the possibilities. When there are uncertainties about the time needs of tasks, or of the possibility of arrival of new tasks, the distributed scheduling problem requires mechanisms to maximize expected performance and to make forecasts about future activities [34]. When there might not be feasible schedules to satisfy all agents, issues arise about how agents should decide which plans to combine to maximize their global performance [17]. More complex representations of reactive plans and techniques for coordinating them based on model-checking and Petri-net-based mechanisms have also been explored [27] [32] [44].

Iterative Plan Formation. Plan merging is a powerful technique for increasing parallelism in the planning process as well as during execution. The synchronization and scheduling algorithms outlined above can be carried out in centralized and decentralized ways, where the flow is generally that of (1) assign goals to agents; (2) agents formulate local plans; (3) local plans are exchanged and combined; (4) messaging and/or timing commitments are imposed to resolve negative plan interactions. The parallels between this method of planning and the task-sharing style of distributed problem-solving should be obvious. But just as we discovered in distributed problem solving, not all problems are like the Tower of Hanoi; sometimes, local decisions are dependent on the decisions of others. This raises the question of the degree to which local plans should be formulated with an eye on the coordination issues, rather than as if the agent could work alone.

One way of tempering proposed local plans based on global constraints is to require agents to search through larger spaces of plans rather than each proposing a single specific plan. Thus, each agent might construct the set of *all* feasible plans for accomplishing its own goals. The distributed planning process then consists of a search through how subsets of agents' plans can fit together.

Ephrati and Rosenschein [16] have developed a plan combination search approach for doing this kind of search, where the emphasis is on beginning with encompassing sets of possible plans and refining these to converge on a nearly optimal subset. They avoid commitment to sequences of actions by specifying sets of propositions that hold as a result of action sequences instead. The agents engage in the search by proposing, given a particular set of propositions about the world, the changes to that set that they each can make with a single action from their plans. These are all considered so as to generate candidate next sets of propositions about the world, and these candidates can be ranked using an A* heuristic (where each agent can use its plans to estimate the cost from the candidate to completing its own goals). The best candidate is chosen and the process repeats, until no agent wants to propose any changes (each has accomplished its goal).

Note that, depending on the more global movement of the plan, an agent will be narrowing down the plan it expects to use to accomplish its own private goals. Thus, agents are simultaneously searching for which local plan to use as well as for synchronization constraints on their actions (since in many cases the optimal step forward in the set of achieved propositions might omit the possible contributions of an agent, meaning that the agent should not perform an action at the time.

An alternative to this approach instead exploits the hierarchical structure of a plan space to perform distributed hierarchical planning. By now, hierarchical planning is well-established in the AI literature. It has substantial advantages (as exemplified in the ToH problem) in that some interactions can be worked out in more abstract plan spaces, thereby pruning away large portions of the more detailed spaces. In the distributed planning literature, the advantages of hierarchical planning were first investigated by Corkill.

Corkill's work considered a distributed version of Sacerdoti's NOAH system. He added a "decompose plan" critic that would look for conjunctive goals to distribute. Thus, in a blocks-world problem (the infamous Sussman's Anomaly, for instance), the initial plan refinement of (AND (ON A B) (ON B C)) leads to a plan network with two concurrent paths, one for each of the conjuncts. The decompose-plan critic gives a copy of the plan network to a second agent, where each of the two agents now represents the goal it is to achieve as well as a parallel node in the network that represents a model of the other agent's plan. Then the agents proceed refine their abstract plans to successively detailed levels. As an agent does so, it can communicate with the other one about the changes that it expects to make to the world state, so that each can separately detect conflicts. For example, when an agent learns that the other is going to make block B not clear (it does not know the details of how) it can determine that this will interfere with stacking B on C, and can ask the first agent to WAIT on the action that causes that change until it has received permission to go on. This process can continue until a synchronized set of detailed plans are formed.

A variation on the hierarchical distributed planning approach is to allow each agent to represent its local planned behaviors at multiple levels of abstraction, any of which can suffice to resolve all conflicts. In this hierarchical behavior-space search approach to distributed planning, the outer loop of the protocol identifies a particular level of abstraction to work with, and whether conflicts should be resolved at this level or passed to more detailed levels. The inner loop of the protocol conducts what can be thought of as a distributed constraint satisfaction search to resolve the conflicts. Because the plans at various abstraction levels dictate the behaviors of agents to a particular degree, this approach has been characterized as search through hierarchical behavior space [12]. The algorithm is now presented (Figure 3):

1. Initialize the current-abstraction-level to the most abstract level.
2. Agents exchange descriptions of the plans and goals of interest at the current level.
3. Remove plans with no potential conflicts. If the set is empty, then done, otherwise determine whether to resolve conflicts at the current level or at a deeper level.
4. If conflicts are to be resolved at a deeper level, set the current level to the next deeper level and set the plans/goals of interest to the refinements of the plans with potential conflicts. Go to step 2.
5. If conflicts are to be resolved at this level:

a. Agents form a total order. Top agent is the current superior.
b. Current superior sends down its plan to the others.
c. Other agents change their plans to work properly with those of the current-superior. Before confirming with the current superior, an agent also doublechecks that its plan changes do not conflict with previous superiors.
d. Once no further changes are needed among the plans of the inferior agents, the current superior becomes a previous superior and the next agent in the total order becomes the superior. Return to step b. If there is no next agent, then the protocol terminates and the agents have coordinated their plans.

Figure 3: Hierarchical Behavior-Space Search Algorithm

Provided that there are finite abstraction levels and that agents are restricted in the changes to their plans that they can make such that they cannot get into cyclic plan generation patterns, the above protocol is assured to terminate. A challenge lies in the outer loop, in terms of deciding whether to resolve at an abstract level or to go deeper. The advantage of resolving a conflict at an abstract level is that it reduces the amount of search, and thus yields coordinated plans with less time and messaging. The disadvantage is that the coordination constraints at an abstract level might impose unnecessary limits on more detailed actions. At more detailed levels, the precise interaction problems can be recognized and resolved, while at abstract levels more inefficient coordination solutions might work. The tradeoffs between long-term, simple, but possibly inefficient coordination decisions versus more responsive but complex run-time coordination decisions is invariably domain-dependent. The goal is to have mechanisms that support the broad spectrum of possibilities.

Negotiation in Distributed Planning. In the above, we considered how agents can determine that conflicts exist between their plans and how to impose constraints on (usually when they take) their actions to avoid conflict. Sometimes, determining which agent should wait for another is fairly random and arbitrary. Exceptions, however, exist. A large amount of work in negotiation is concerned with these issues, so we only touch on them briefly here.

Sometimes the selection of the agent that should revise its local plans is based on models of the possibilities open to the agents. For example, Steeb and Cammarata, in the air-traffic control domain, were concerned with which of the various aircraft should alter direction to decrease potentially dangerous congestion. Their agents exchanged descriptions indicating their flexibility, and the agent that had the most other options was asked to change its plan, in an early DAI application of the least-constrained agent heuristic (see associated prior sections on constrained search and constraint satisfaction).

Of course, these and other negotiation mechanisms for resolving goals presume that agents are honest about the importance of their goals and their options for how to achieve them. Issues of how to encourage self-interested agents to be honest are covered elsewhere [43]. However, clearly agents have self-interest in looking for opportunities to work to their mutual benefit by accomplishing goals that each other need. However, although the space of possible conflicts between agents is large, the space of possible cooperative activities can be even larger, and introduces a variety of utility assessments. That is, while it can be argued that agents that have conflicts always should resolve them (since the system might collapse if conflicts are manifested), the case for potential cooperative actions is not so strong. Usually, cooperation is "better," but the degree to which agents benefit might not outweigh the efforts they expend in finding cooperative opportunities. Thus, work on distributed planning that focuses on planning for mutually beneficial actions even though they were not strictly necessary has been limited to several forays into studies within well-defined boundaries. For example, partial global planning (see next subsection) emphasized a search for generating partial solutions near partial solution boundaries with other agents, so as to provide them with useful focusing information early on (see prior section on communication strategies). The work of von Martial [36] concentrated on strategies that agents can use to exploit "favor relations" among their goals, such as accomplishing a goal for another agent while pursuing its own goal.

6 Distributed Planning and Execution

Of course, distributed planning does not occur in a vacuum. The product of distributed planning needs to be executed. The relationships between planning and execution are an important topic in AI in general, and the added complexity of coordinating plans only compounds the challenges. In this section, we consider strategies for combining coordination, planning, and execution.

6.1 Post-planning Coordination

The distributed planning approach based on plan merging essentially sequentialized the processes in terms of allowing agents to plan, then coordinating the plans, and then executing them. This is reasonable approach given that the agents individually build plans that are likely to be able to be coordinated, and that the coordinated result is likely to executed successfully. If, during execution, one (or more) plans for agents fail to progress as expected, the coordinated plan set is in danger of failing as a whole.

As in classical planning systems, there are several routes of recourse to this problem. One is contingency planning. Each agent formulates not only its expected plan, but also alternative (branches of) plans to respond to possible contingencies that can arise at execution time. These larger plans, with their conditional branches, can then be merged and coordinated. The coordination process of course is more complicated because of the need to consider the various combinations of plan execution threads that could be pursued. By annotating the plan choices with the conditions, a more sophisticated coordination process can ignore combinations of conditional plans whose conditions cannot be satisfied in the same run.

A second means of dealing with dynamics is through monitoring and replanning: Each agent monitors its plan execution, and if there is a deviation it stops all agents' progress, and the plan-coordinate-execute cycle is repeated. Obviously, if this happens frequently, a substantial expenditure of effort for planning and coordination can result. Sometimes, strategies such as repairing the previous plans, or accessing a library of reusable plans [49] can reduce the effort to make it managable.

Significant overhead can of course be saved if a plan deviation can be addressed locally rather than having to require coordination. For example, rather than coordinating sequences of actions, the agents might coordinate their plans at an abstract level. Then, during execution, an agent can replan details without requiring coordination with others so long as its plan revision fits within the coordinated abstract plan. This approach has been taken in the team plan execution work of Kinney and colleagues, for example [29]. The perceptive reader will also recognize in this approach the flavor of organizational structuring and distributed planning in a hierarchical behavior space: so long as it remains within the scope of its roles and responsibilities, an agent can individually decide what is the best way of accomplishing its goals. By moving to coordinate at the most abstract plan level, the process essentially reverses from post-planning to pre-planning coordination.

6.2 Pre-planning Coordination

Before an agent begins planning at all, can coordination be done to ensure that, whatever it plans to do, the agent will be coordinated with others? The answer is of course yes, assuming that the coordination restrictions are acceptable. This was the answer in organizational structuring in distributed problem solving, where an agent could choose to work on any part of the problem so long as it fit within its range of responsibilities.

A variation on this theme is captured in the work on social laws [46]. A social law is a prohibition against particular choices of actions in particular contexts. For example, entering an intersection on a red light is prohibited, as might *not* entering the intersection on a green light. These laws can be derived by working from undesirable states of the world backwards to find combinations of actions that lead to those states, and then imposing restrictions on actions so that the combinations cannot arise. A challenge is to find restrictions that prevent undesirable states without handcuffing agents from achieving states that are acceptable and desirable. When overly constrictive, relaxations of social laws can be made [3].

Alternatively, in domains where conflict avoidance is not a key consideration, it is still possible that agents might mutually benefit if they each prefer to take actions that benefit society as a whole, even if not directly relevant to the agent's goal. For example, in a Distributed Delivery application, it could be that a delivery agent is passing by a location where an object is awaiting pickup by a different agent. The agent passing by could potentially pick up the object and deliver it itself, or deliver it to a location along its route that will be a more convenient pickup point for the other agent. For example, the delivery agents might pass through a "hub" location. The bias toward doing such favors for other agents could be encoded into cooperative state-changing rules [23] that require agents to take such cooperative actions even to their individual detriment, as long as they are not detrimental beyond some threshold.

6.3 Continual Distributed Planning and Execution

More generally, between approaches that assume agents have detailed plans to coordinate and approaches that assume general-purpose coordination policies can apply to all planning situations, lies work that is more flexible about at what point between the most abstract and most detailed plan representations different kinds of coordination should be done. Perhaps the search for the proper level is conducted through a hierarchical protocol, or perhaps it is predefined. In either case, planning and coordination are interleaved with each other, and often with execution as well.

For example, Partial Global Planning [11] assumes that agents will interleave actions with planning, and that in turn planning and coordination will occur asynchronously. At any given time, an agent might be taking an action based on its most recently updated plan, but that plan in turn might still be in the process of being coordinated with the plans of other agents. Viewed as a distributed problem of deciding on actions in an evolving space, we can characterize Partial Global Planning in the following way:

Task Decomposition - Partial Global Planning starts with the premise that tasks are inherently decomposed -- or at least that they could be. Therefore, unlike planning techniques that assume that the overall task to be planned for is known by one agent, partial global planning assumes that an agent with a task to plan for might be unaware at the outset as to what tasks (if any) other agents might be planning for, and how (and whether) those tasks might be related to its own as in the DVM task.

Local plan formulation - Before an agent can coordinate with others using Partial Global Planning, it must first develop an understanding of what goals it is trying to achieve and what actions it is likely to take to achieve them. Since most agents will be concurrently concerned with multiple goals, local plans will most often be uncertain, involving branches of alternative actions depending on the results of previous actions and changes in the environmental context in carrying out the plan.

Local plan abstraction - Abstraction plays a key role in coordination, since coordination that is both correct and computationally efficient requires that agents have models of themselves and others that are only detailed enough to gainfully enhance collective performance. In Partial Global Planning, for example, agents are designed only to reveal their major plan steps that could be of interest to other agents.

Communication - In Partial Global Planning, the knowledge to guide communication is contained in the *Meta-Level Organization* (MLO), which specifies who needs to know the plans of a particular agent, and who has authority to impose new plans on an agent based on having a more global view.

Partial global plan construction and modification- Local plans that can be seen as contributing to a single partial global goal can be integrated into a partial global plan, which captures the planned concurrent activities (at the abstract plan step level) of the individuals. By analyzing these activities, an agent that has constructed the partial global plan can identify opportunities for improved coordination such as facilitating task achievement of others by performing related tasks earlier, and of avoiding redundant task achievement.

Communication planning - After reordering the major local plan steps of the participating agents so as to yield a more coordinated plan, interactions, in the form of communicating the results of tasks, are also planned. By examining the partial global plan, an agent can determine when a task will be completed by one agent that could be of interest to another agent, and can explicitly plan the communication action to transmit the result.

Acting on partial global plans - Once a partial global plan has been constructed and the concurrent local and communicative actions have been ordered, these activities must be translated back to the local level so that they can be carried out.

Ongoing modification - A challenge in coordination is deciding when the changes in local plans are significant enough to warrant communication and recoordination. The danger in being too sensitive to changes is that an agent that informs others of minor changes can cause a chain reaction of minor changes, where the slight improvement in coordination is more than offset by the effort spent in getting it. On the other hand, being too insensitive can lead to very poor performance, as agents' local activities do not mesh well because each is expecting the other to act according to the partial global plan, which is not being followed very closely anymore. In PGP, a system designer has the ability to specify parametrically the threshold that defines significant temporal deviation from planned activity.

Task reallocation - In some circumstances, the exogenous task decomposition and allocation might leave agents with disproportionate task loads. Through PGP, agents that exchange abstract models of their activities will be able to detect whether they are overburdened, and candidate agents that are underburdened. By generating and proposing partial global plans that represent others taking over some of its tasks, an agent essentially suggests a contracting relationship among the agents, which could be adopted.

The space of strategies for engaging in continual planning (planning interleaved with execution) among multiple agents was also the subject of a special issue of the AI Magazine [1]. In what follows, we briefly overview some of the examples of distributed continual planning systems highlighted in that issue. As is often the case, the distinctions between the systems tend to hinge on assumptions about such things as whether the agents can count on a global view in the creation of their plan, and whether they are working toward a single goal or multiple goals.

For example, the work of Myers [37] concentrates on a multiagent system for planning and execution, where the overall system forms, executes, monitors, and repairs a single plan through the cooperative interactions of agents that are dedicated to each of these (and other) aspects of planning. Similarly, desJardins and Wolverton [9] focus on how agents which are each experts at portions of the planning needs can work together and merge their results to form a coherent, complex plan. Boutilier [2] describes means for formulating plans that maximize expected collective performance, taking into account the execution-time capabilities of agents to communicate and coordinate.

Other work emphasizes managing and executing plans in that are being conducted in a multiagent environment. Pollack and Horty [40] concentrates on how an agent makes plans and manages commitments to plans during execution despite changes to the environment and in the multiagent context. Grosz and colleagues [24], Tambe and Jung [51], and Durfee [15] take a more collective "team" perspective, studying how agents that are supposed to be collaborating (and know they are) can decide on responsibilities, can balance the need to support team commitments with allowing some degree of extemporaneous execution, and can formally ensure that their individual and group intentions interweave to accomplish their shared purpose.

6.4 Runtime Plan Coordination Without Communication

While tailored for dynamic domains, PGP still assumes that agents can and will exchange planning information over time to coordinate their actions. In some applications, however, runtime recoordination needs to be done when agents cannot or should not communicate. We briefly touch on plan coordination mechanisms for such circumstances.

One way of coordinated without explicit communication is to allow agents to infer each others plans based on observations. The plan recognition literature focuses on how observed actions can lead to hypotheses about the plans being executed by others. While generally more uncertain than coordination using explicit communication, observation-based plan coordination can still achieve high-quality results and, under some circumstances can outperform communication-based distributed planning [25].

Another way of coordinating without explicit communication is to allow agents to make inferences about the choices others are likely to make based on assumptions about their rationality [42] or about how they view the world. For example, if Distributed Delivery agents are going to hand off objects to each other, they might infer that some locations (such as a hub) are more likely to be mutually recognized as good choices. Such solutions to choice problems have been referred to as focal points [19].

7 Conclusions

Distributed planning has a variety of reasonably well-studied tools and techniques in its repertoire. One of the important challenges to the field is in characterizing these tools and undertanding where and when to apply each. To some extent, the lack of specificity in the term "distributed planning" in terms of whether the process or the product or both of planning is distributed has hampered communication within the field, but more fundamental issues of articulating the foundational assumptions behind different approaches still need to be addressed. Until many of the assumed context and semantics for plans are unveiled, the goal of having heterogeneous plan generation and plan execution agents work together is likely to remain elusive.

The field of distributed problem solving is even more wide open, because the characterization of a "problem" is that much broader. As we have tried to emphasize, distributed plan formation and, in many cases, execution can be thought of as distributed problem solving tasks. Representations and general-purpose strategies for distributed problem solving are thus even more elusive. In this paper we have characterized basic classes of strategies such as task-sharing and result-sharing. Ultimately, the purpose of any strategy is to share the right information about tasks, capabilities, availabilities, partial results, or whatever so that each agent is doing the best thing that it can for the group at any given time. Of course, exchanging and using the information that renders such choices can itself be costly, and opens the door to misinterpretation that makes matters worse rather than better. All of these considerations factor into the definition and implementation of a distributed problem strategy, but formulating such a strategy is still has more "art" to it than we like to see in an engineering discipline.

References

1. *AI Magazine*, 20(4), Winter 1999. Special issue on Distributed Continual Planning.
2. Craig Boutilier. "Multiagent Systems: Challenges and Opportunities for Decision-Theoretic Planning." AI Magazine 20(4):35-43, Winter 1999.
3. Will Briggs and Diane J. Cook. Flexible social laws. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), August 1995.
4. Susan E. Conry, Kazuhiro Kuwabara, Victor R. Lesser, and Robert A. Meyer. Multistage negotiation for distributed constraint satisfaction. IEEE Trans. of Systems, Man, and Cybernetics SMC-21(6):1462-1477, Nov. 1991.
5. Daniel D. Corkill. A Framework for Organizational Self-Design in Distributed Problem Solving Networks. PhD thesis, University of Massachusetts, December 1982.
6. Randall Davis and Reid Smith. Negotiation as a metaphor for distributed problem solving. Artificial Intelligence 20:63-109, 1983.
7. Keith Decker and Victor Lesser. A one-shot dynamic coordination algorithm for distributed sensor networks. Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), pages 210-216, July 1993.
8. Keith Decker and Victor Lesser. Designing a family of coordination mechanisms. Proceedings of the First International Conf. on Multi-Agent Systems (ICMAS-95), pages 73-80, June 1995.
9. Marie desJardins and Michael Wolverton. "Coordinating a Distributed Planning System." AI Magazine 20(4):45-53, Winter 1999.
10. Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. "Cooperation Through Communication in a Distributed Problem Solving Network," Chapter 2 in M. Huhns (ed.) Distributed Artificial Intelligence , Pitman 1987.
11. Edmund H. Durfee. Coordination of Distributed Problem Solvers, Kluwer Academic Press, Boston 1988.
12. Edmund H. Durfee and Thomas A. Montgomery. "Coordination as Distributed Search in a Hierarchical Behavior Space." IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Artificial Intelligence, SMC-21(6):1363-1378, November 1991.
13. Edmund H. Durfee, Patrick G. Kenny, and Karl C. Kluge. Integrated Premission Planning and Execution for Unmanned Ground Vehicles. Proceedings of the First International Conference on Autonomous Agents, pages 348-354, February 1997.

14. Edmund H. Durfee. “Distributed Problem Solving and Planning.” Chapter 3 in [52].
15. Edmund H. Durfee. “Distributed Continual Planning for Unmanned Ground Vehicle Teams.” AI Magazine 20(4):55-61, Winter 1999.
16. Eithan Ephrati and Jeffrey S. Rosenschein. Divide and conquer in multi-agent planning. Proceedings of the Twelfth National Conf. on Artificial Intelligence (AAAI-94), pages 375-380, July 1994.
17. Eithan Ephrati, Martha E. Pollack, and Jeffrey S. Rosenschein. A tractable heuristic that maximizes global utility through local plan combination. Proceedings of the First International Conf. on Multi-Agent Systems (ICMAS-95), pages 94-101, June 1995.
18. R. D. Fennell and V. R. Lesser. Parallelism in AI problem solving: A case study of HEARSAY-II. IEEE Trans. on Computers C-26(2):98-111, 1977.
19. Maier Fenster, Sarit Kraus, and Jeffrey S. Rosenschein. Coordination without communication: experimental validation of focal point techniques. Proceedings of the First International Conf. on Multi-Agent Systems (ICMAS-95), pages 102-108, June 1995.
20. Michael Fisher and Michael Wooldridge. Distributed problem-solving as concurrent theorem-proving. Proceedings of MAAMAW’97, Lecture notes in Artificial Intelligence, Springer-Verlag.
21. Les Gasser. “Computational Organization Theory.” Chapter 7 of [52].
22. Michael Georgeff. Communication and Interaction in multi-agent planning. Proceedings of the Third National Conf. on Artificial Intelligence (AAAI-83), pages 125-129, July 1983.
23. Claudia Goldman and Jeffrey S. Rosenschein. Emergent coordination through the use of cooperative state-changing rules. Proceedings of the Twelfth National Conf. on Artificial Intelligence (AAAI-94), pages 408-413, July 1994.
24. Barbara J. Grosz, Luke Hunsberger, and Sarit Kraus. “Planning and Acting Together.” AI Magazine 20(4):23-34, Winter 1999.
25. Marcus J. Huber and Edmund H. Durfee. An initial assessment of plan-recognition-based coordination for multi-agent teams. Proceedings of the Second International Conf. on Multi-Agent Systems (ICMAS-96), pages 126-133, December 1996.
26. Toru Ishida, Les Gasser, and Makoto Yokoo. Organization self-design of distributed production systems, IEEE Trans on Knowl and Data Sys DKE4(2):123-134.
27. Froduald Kabanza. Synchronizing multiagent plans using temporal logic specifications. Proceedings of the First International Conf. on Multi-Agent Systems (ICMAS-95), pages 217-224, June 1995.
28. Subbarao Kambhampati, Mark Cutkosky, Marty Tenenbaum, and Soo Hong Lee. Combining specialized reasoners and general purpose planners: A case study. Proceedings of the Ninth National Conference on Artificial Intelligence, pages 199-205, July 1991.
29. David Kinney, Magus Ljungberg, Anand Rao, Elizabeth Sonenberg, Gil Tidhar, and Eric Werner, "Planned Team Activity," Preproceedings of the Fourth European Workshop on Modeling Autonomous Agents in a MultiAgent World, July 1992.
30. Susan E. Lander and Victor R. Lesser. Understanding the role of negotiation in distributed search among heterogeneous agents. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93) , pages 438-444, August 1993.
31. Amy L. Lansky. Localized Search for Controlling Automated Reasoning. Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control, pages 115-125, November 1990.
32. Jaeho Lee. An Explicit Semantics for Coordinated Multiagent Plan Execution. PhD dissertation. University of Michigan, 1997.
33. Victor R. Lesser and Daniel D. Corkill. Functionally accurate, cooperative distributed systems. IEEE Trans. on Systems, Man, and Cybernetics SMC-11(1):81-96, 1981.
34. Jyi-Shane Liu and Katia P. Sycara. Multiagent coordination in tightly coupled task scheduling. Proceedings of the Second International Conf. on Multi-Agent Systems (ICMAS-96), pages 181-188, December 1996.

35. Douglas MacIntosh, Susan Conry, and Robert Meyer. Distributed automated reasoning: Issues in coordination, cooperation, and performance. IEEE Trans. on Systems, Man, and Cybernetics SMC-21(6):1307-1316.
36. Frank von Martial. Coordinating Plans of Autonomous Agents. Lecture notes in Artificial Intelligence, Springer-Verlag, 1992.
37. Karen L. Myers. "CPEF: A Continuous Planning and Execution Framework." AI Magazine 20(4):63-69, Winter 1999.
38. R. Neches, R. Fikes, T. Finin, R. Gruber, R. Patil, T. Senator, and W. Swartout (1991). "Enabling Technology for Knowledge Sharing." AI Magazine 12(3): 36-56.
39. H. Edward Pattison, Daniel D. Corkill, and Victor R. Lesser. Instantiating descriptions of organizational structures. In M. Huhns (ed.) Distributed Artificial Intelligence. London, Pittman.
40. Martha E. Pollack and John F. Horty. "There's More to Life Than Making Plans: Plan Management in Dynamic Multiagent Environments." AI Magazine 20(4):71-83, Winter 1999.
41. M. V. Nagendra Prasad, Keith Decker, Alan Garvey, and Victor Lesser. Exploring organizational designs with TAEMS: A case study of distributed data processing. Proceedings of the Second International Conf. on Multi-Agent Systems (ICMAS-96), pages 283-290, December 1996.
42. Jeffrey S. Rosenschein and John S. Breese. Communication-free interactions among rational agents: A probabilistic approach. In Gasser and Huhns (eds.) Distributed Artificial Intelligence volume II, pages 99-118, Morgan Kaufmann Publishers.
43. Tuomas Sandholm. "Distributed Rational Decision Making." Chapter 5 in [52].
44. Amal El Fallah Seghrouchni and Serge Haddad. A recursive model for distributed planning. Proceedings of the Second International Conf. on Multi-Agent Systems (ICMAS-96), pages 307-314, December 1996.
45. Sandip Sen and Edmund H. Durfee. A contracting model for flexible distributed scheduling. Annals of Operations Research, vol. 65, pp. 195-222, 1996.
46. Yoav Shaham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies. Proceedings of the Tenth National Conf. on Artificial Intelligence (AAAI-92), pages 276-281-380, July 1992.
47. Young-pa So and Edmund H. Durfee. Designing tree-structured organizations for computational agents. Computational and Mathematical Organization Theory 2(3):219-246, Fall 1996.
48. John A. Stankovic, Krithi Ramamritham, and S.-C. Cheng. Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems. IEEE Trans. on Computers C-34(12):1130-1143, 1985.
49. Toshiharu Sugawara. Reusing past plans in distributed planning. Proceedings of the First International Conf. on Multi-Agent Systems (ICMAS-95), pages 360-367, June 1995.
50. Katia Sycara, Steven Roth, Norman Sadeh, and Mark Fox. Distributed constrained heuristic search. IEEE Transactions on Systems, Man, and Cybernetics SMC-21(6):1446-1461.
51. Milind Tambe and Hyuckchul Jung. "The Benefits of Arguing in a Team." AI Magazine 20(4):85-92, Winter 1999.
52. Gerhard Weiss, editor. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, Cambridge MA, 1999.
53. Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. Journal of Artificial Intelligence Research , 1:1-23, 1993.
54. Keith J. Werkman. Multiple agent cooperative design evaluation using negotiation. Proceedings of the Second International Conference on Artificial Intelligence in Design, Pittsburgh PA, June 1992.

55. D. E. Wilkins and K. L. Myers. "A common knowledge representation for plan generation and reactive execution." Journal of Logic and Computation, 5(6):731-761, 1995.
56. Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. "Distributed Constraint Satisfaction Problem: Formalization and Algorithms." IEEE Transactions on Knowledge and Data Engineering, TKDE10(5):673-685, September 1998.
57. Makoto Yokoo and Toru Ishida. "Search Algorithms for Agents." Chapter 4 in [52].
58. Chenqi Zhang. Cooperation under uncertainty in distributed expert systems. Artificial Intelligence 56:21-69, 1992.

Automated Negotiation and Decision Making in Multiagent Environments[*]

Sarit Kraus

[1] Dept. of Mathematics and Computer Science
Bar-Ilan University
Ramat-Gan, 52900 Israel
[2] Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742, USA
sarit@umiacs.umd.edu
http://www.umiacs.umd.edu/users/sarit

Abstract. This paper presents some of the key techniques for reaching agreements in multi-agent environments. It discusses game-theory and economics based techniques: strategic negotiation, auctions, coalition formation, market-oriented programming and contracting. It also presents logical based mechanisms for argumentations. The focus of the survey is on negotiation of self-interested agents, but several mechanisms for cooperative agents who need to resolve conflicts that arise from conflicting beliefs about different aspects of their environment are also mentioned. For space reasons, we couldn't cover all the relevant works, and the papers that are mentioned only demonstrate the possible approaches. We present some of the properties of the approaches using our own previous work.

1 Introduction

Negotiation has been a subject of central interest in multi-agent systems, as it has been in economics and political science. The word has been used in a variety of ways, though in general it refers to communication processes that further coordination and cooperation. Negotiations can be used to resolve conflicts in a wide variety of multi-agent domains [28]. Examples of such applications include conflicts over the usage of joint resources or task assignments, conflicts concerning document allocation in multi-server environments and conflicts between a buyer and a seller in electronic commerce.

When building an autonomous agent which is capable of flexible and sophisticated negotiation, the main questions that should be considered are: (i) what

[*] This work is supported in part by NSF under grant No. IIS-9820657. This paper is based on chapters in S. Kraus. Strategic Negotiation in Multi-Agent Environments, MIT Press, Cambridge, USA, 2001 and on S. Kraus, K. Sycara and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation, *Artificial Intelligence Journal*, 104(1-2):1-69, 1998.

negotiation protocol will be used? (ii) what reasoning model, decision making procedures and strategies will the agents employ?

Several protocols for auctions, strategic negotiation and coalition formation are considered and we discuss their applicability in various multi-agent domains. We will present formal models for agent reasoning and we will discuss methods for identifying strategies for agents interacting using a specific protocol.

Evaluation of the results of multi-agent protocols is not an easy task. Since the agents are self-interested, when saying, for example, a "negotiation was successful" the question "successful for whom?" must be asked, since each agent is concerned only about its own benefits or losses from the resolution of the negotiation. Nevertheless, there are certain parameters that can be used to evaluate different protocols.

Negotiation Time: Negotiations which end without delay are preferred over negotiations which are time-consuming.
It is assumed that a delay in reaching an agreement causes an increase in the cost of communication and computation time spent on the negotiation. We want to prevent the agents from spending too much time on negotiations resulting in deviation from their timetables for satisfying their goals.

Efficiency: An efficient outcome of the negotiations is preferred. In other words, an outcome that increases the number of agents which will be satisfied by the negotiation results and the agents' satisfaction levels from the negotiation results.
Thus, it is preferred that the agents reach *Pareto optimal* agreements[1] In addition, if there is an agreement that is better for all the agents than opting out, then it is preferred that the negotiations will end with an agreement.

Simplicity: Negotiation processes that are simple and efficient are better than complex processes. Being a "simple strategy" means that it is feasible to build it into an automated agent. A "simple strategy" also presumes that an agent will be able to compute the strategy in a reasonable amount of time.

Stability: A set of negotiation strategies are stable if, given that all the other agents included in the set are following their strategies, it is beneficial to an agent to follow its strategy too. Negotiation protocols which have stable strategies are more useful in multiagent environments than protocols which are unstable. If there are stable strategies, we can recommend to **all** agent designers to build the relevant strategies into their agents. No designer will benefit by building agents that use any other strategy.

Money transfer: Money transfer may be used to resolve conflicts. For example, a server may "sell" a data item to another server when relocating this item. This can be done by providing the agents with a monetary system and with a mechanism for secure payments. Since maintaining such a monetary system requires resources and efforts, negotiation protocols that do not require money transfers are preferred.

[1] An agreement is Pareto optimal if there is no other agreement that dominates it, i.e., there is no other agreement that is better for some of the agents and not worse for the others.

The remainder of this paper is structured as follows. In the next section we will present a short survey of negotiation approaches in Distributed Artificial Intelligence (DAI) and in social sciences. Then we will discuss the strategic-negotiation model (section 2.3) which is based on the game-theory model of bargaining with alternating offers. Auctions are discussed in section 3, and market-oriented programming is briefly discussed in section 4. Agents can also cooperate by forming coalitions. This form of cooperation is presented in section 5. Contracting which is another form of reaching cooperation is surveyed in section 6. Finally, logical approaches to negotiation are presented in section 7.

2 Negotiation Models

We will first present a short survey of various negotiation approaches in Distributed Artificial Intelligence. Then, we will describe the two main approaches to negotiations in the social sciences and will demonstrate the application of one of the approaches to multiagent systems.

2.1 Negotiation Models in DAI

Negotiations were used in DAI both in Distributed Problem Solving (DPS) where the agents are cooperative and in Multiagent Systems (MA) where the agents are self-interested. Several works in DPS use negotiation for distributed planning and distributed search for possible solutions for hard problems. For example, Conry et al. [10] suggest multi-stage negotiation to solve distributed constraint satisfaction problems when no central planner exists. Moehlman and Lesser [52] use negotiation as a tool for distributed planning: each agent has certain important constraints, and it tries to find a feasible solution using a negotiation process. They applied this approach in the Phoenix fireman array. Lander and Lesser [44] use a negotiation search, which is a multi-stage negotiation as a means of cooperation while searching and solving conflicts among the agents.

For the MA environments, Rosenschein and Zlotkin [73] identified three distinct domains where negotiation is applicable and found a different strategy for each domain: (i) Task-Oriented Domain: Finding ways in which agents can negotiate to come to an agreement, and allocating their tasks in a way that is beneficial to everyone; (ii) State-Oriented Domain: Finding actions which change the state of the "world" and serve the agents' goals; and (iii) Worth-Oriented Domain: Same as (ii) above, but, in this domain, the decision is taken according to the maximum utility the agents gain from the states.

Sycara [100,99] presented a model of negotiation that combines case-based reasoning and optimization of multi-attribute utilities. In her work agents try to influence the goals and intentions of their opponents. Kraus and Lehmann [39] developed an automated Diplomacy player that negotiates and plays well in actual games against human players. Sierra et al. [94] present a model of negotiation for autonomous agents to reach agreements about the provision of service by one agent to another. Their model defines a range of strategies and tactics, distilled from intuition about good behavioral practice in human negotiation, that agents can employ to generate offers and evaluate proposals. Zeng and Sycara [116] consider negotiation in a marketing environment with a learning process in which the buyer and the seller update their beliefs about the opponent's reservation price[2] using the Bayesian rule. Sandholm and Lesser [85] discuss issues, such as levels of commitment, that arise in automated negotiation among self-interested agents whose rationality is bounded by computational complexity.

2.2 Negotiation Approaches in the Social Sciences

In social sciences there are two main approaches to the development of theories relating to negotiation. The first approach is the formal theory of bargaining e.g., [75,62], constituting a formal, game-theoretic approach that provides clear analyses of various situations and precise results concerning the strategy a negotiator should choose. However, this approach can only be applied to situations satisfying very restricted assumptions. In particular, this approach assumes that the agents are acting rationally, have large computation capabilities and follow strict negotiation protocols.

The second approach, which we refer to as the negotiation guides approach, comprises informal theories which attempt to identify possible general beneficial strategies for a negotiator. The works based on this approach advise a negotiator how to behave in order to reach beneficial results in a negotiation (see, for example, [68,13,11,32,29,22]). These negotiation guides do not presuppose the strong restrictions and assumptions presented in the game-theoretic models. Applying these methods to automated systems is more difficult than using the first approach, since there are neither formal theories nor strategies that can be used.[3] In the next section we demonstrate the application of the formal approach to multiagent systems.

2.3 Strategic Negotiation

The strategic-negotiation model is based on Rubinstein's model of alternating offers [77]. In the strategic model there are N agents, $\mathbf{Agents} = \{A_1, ..., A_N\}$. The agents need to reach an agreement on a given issue. It is assumed that the agents can take actions in the negotiation only at certain times in the set $\mathcal{T} = \{0, 1, 2...\}$ that are determined in advance and are known to the agents.

In each period $t \in \mathcal{T}$ of the negotiation, if the negotiation has not terminated earlier, an agent whose turn it is to make an offer at time t, will suggest a possible

[2] The reservation price of the seller is the price below which the seller refuses to sell. The reservation price of the buyer is the price above which the buyer refuses to buy.

[3] These methods can be used in domains where people interact with each other and with automated systems, and situations where automated systems interact in environments without predefined regulations. These informal models can serve as guides for the development of negotiation heuristics [39] or as a basis for the development of a logical negotiation model [40].

agreement (with respect to the specific negotiation issue), and each of the other agents may either accept the offer (choose Yes), reject it (choose No), or opt out of the negotiation (choose Opt). If an offer is accepted by all the agents (i.e., all of them choose Yes), then the negotiation ends, and this offer is implemented. If at least one of the agents opts out of the negotiation, then the negotiation ends and a conflictual outcome results. If no agent has chosen “Opt,” but at least one of the agents has rejected the offer, the negotiation proceeds to period $t+1$, and the next agent makes a counteroffer, the other agents respond, and so on. We assume that an agent responding to an offer is not informed of the other responses during the current negotiation period. We call this protocol a *simultaneous response* protocol.[4] $j(t)$ will denote the agent that makes an offer at time period t. The following example demonstrate these notions.

Example 1 (Data Allocation in Large Databases). There are several information servers, in different geographical areas. Each server stores data, which has to be accessible by clients not only from its geographical area but also from other areas. The topics of interest of each client change dynamically over time, and the set of clients may also change over time. Periodically, new data arrive at the system, and have to be located at one of the servers in the distributed system.

Each server is independent and has its own commercial interests. The servers would like to cooperate with each other in order to make more information available to their clients. Since each server has its own preferences regarding possible data allocations, its interests may conflict with the interests of some of the other servers.

A specific example of a distributed information system is the Data and Information System component of the Earth Observing System (EOSDIS) of NASA [56]. It is a distributed system which supports archival data and distribution of data at multiple and independent data centers (called DAACs). The current policy for data allocation in NASA is static: each DAAC specializes in some topics. When new data arrive at a DAAC, the DAAC checks if the data is relevant to one of its topics, and, if so, it uses criteria, such as storage cost, to determine whether or not to accept the data and store them in its database. The DAAC communicates with other DAACs in those instances in which the data item encompasses the topics of multiple DAACs, or when a data item presented to one DAAC is clearly in the jurisdiction of another DAAC, and then a discussion takes place among the relevant DAAC managers. However, this approach does not take into consideration the location of the information clients, and this may cause delays and transmission costs if data items are stored far from their potential users. Moreover, this method can cause rejection of data items if they do not fall within the criteria of any DAAC, or if they fall under the criteria of a DAAC which cannot support this new product because of budgetary problems.

In this example the agents negotiate to reach an agreement that specifies the location of *all* the relevant data items. In the first time period, the first server

[4] A sequential protocol is considered in [14]. In this protocol an agent responding to an offer is informed of the responses of the preceding agents (assuming that the agents are arranged in a specific order).

offers an allocation, and the other agents either accept the offer, reject it or opt out of the negotiation. If an offer is accepted by all the agents, then the negotiation ends and the proposed allocation is implemented. If at least one of the agents opts out of the negotiation, then a predefined *conflict allocation* is implemented, as described in [87]. If no agent has chosen "Opt," but at least one of the agents has rejected the offer, the negotiation proceeds to the next time period and another agent proposes an allocation, the other agents respond, and so on.

In the strategic-negotiation model there are no rules which bind the agents to any specific strategy. We do not make any assumptions about the offers the agents make during the negotiation. In particular, the agents are not bound to any previous offers that have been made. After an offer is rejected, an agent whose turn it is to suggest a new offer can decide whether to make the same offer again, or to propose a new offer. The protocol only provides a framework for the negotiation process and specifies the termination condition, but there is no limit on the number of periods.

A fair and reasonable method for deciding on the order in which agents will make offers is to arrange them randomly in a specific order before the negotiation begins.[5] That is, the agents will be denoted randomly $A_1, .., A_N$. At each time t, j(t) will be A_i where i is equal to $(t \ mod \ N) + 1$.

The set of possible agreements is denoted $\mathcal{S}$. An outcome of the negotiation may be that an agreement $s \in \mathcal{S}$ will be reached at time $t \in \mathcal{T}$. This outcome is denoted by a pair (s, t). When one of the agents opts out of the negotiations at time period $t \in \mathcal{T}$, the outcome is denoted $(\mathbf{Opt}, t)$. For example, in the data allocation scenario (example 1), an agreement is an allocation which assigns each data item to one of the servers. In this case $\mathcal{S}$ is the set of all possible allocations. The symbol **Disagreement** indicates a perpetual disagreement, i.e., the negotiation continues forever without reaching an agreement and without any of the agents opting out.

The agents' *time preferences* and the preferences between agreements and opting out are the driving force of the model. They will influence the outcome of the negotiation. In particular, agents will not reach an agreement which is not at least as good as opting out for all of them. Otherwise, the agent which prefers opting out over the agreement, will opt out.

Negotiation Strategies An agent's negotiation strategy specifies for the agent what to do next, for each sequence of offers $s_0, s_2, s_3, ..., s_t$. In other words, for the agent whose turn it is to make an offer, it specifies which offer to make next. That is, it indicates to the agent which offer to make at $t + 1$, if in periods 0 until t the offers $s_0, ..., s_t$ had been made and were rejected by at least one of the agents, but none of them has opted out. Similarly, in time periods when it is the agent's turn to respond to an offer, the strategy specifies whether to accept

[5] A distributed algorithm for randomly ordering the agents can be based on the methods of [6].

the offer, reject it or opt out of the negotiation. A strategy profile is a collection of strategies, one for each agent [63].

Subgame Perfect Equilibria The main question is how a rational agent will choose its negotiation strategy. A useful notion is the Nash Equilibrium [57,47] which is defined as follows:

Definition 1 (Nash Equilibrium). *A strategy profile $F = \{f_1, ..., f_N\}$ is a* Nash equilibrium *of a model of alternating offers, if each agent A_i does not have a different strategy yielding an outcome that it prefers to that generated when it chooses f_i, given that every other agent A_j chooses f_j. Briefly, no agent can profitably deviate, given the actions of the other agents.*

This means, that if all the agents use the strategies specified for them in the strategy profile of the Nash equilibrium, then no agent has a motivation to deviate and use another strategy. However, the use of Nash equilibrium in a model of alternating-offers leads to an absurd Nash equilibria [102]: an agent may use a threat that would not be carried out if the agent were put in the position to do so, since the threat move would give the agent lower payoff than it would get by not taking the threatened action. This is because Nash equilibrium strategies may be in equilibrium only in the beginning of the negotiation, but may be unstable in intermediate stages. The concept of subgame perfect equilibrium (SPE) [63], which is a stronger concept, is defined in the following definition and will be used in order to analyze the negotiation.

Definition 2 (Subgame perfect equilibrium:). *A strategy profile is a* subgame perfect equilibrium *of a model of alternating offers if the strategy profile induced in every subgame is a Nash equilibrium of that subgame.*

This means that at any step of the negotiation process, no matter what the history is, no agent has a motivation to deviate and use any strategy other than that defined in the strategy profile.

In situations of incomplete information there is no proper subgame. The *sequential equilibrium* [42], which takes the beliefs of the agents into consideration, can be used in the incomplete information situations.

Example 2. The application of the strategic-negotiation model to the data allocation problem of example 1 was presented in [3] . Using this model, the servers have simple and stable negotiation strategies that result in efficient agreements without delays. It was shown that these methods yield better results than the static allocation policy currently used in EOSDIS (see Example 1).

In particular, it was shown that when servers negotiate to reach an agreement on the allocation of data items and they have complete information various agreements can be reached. It was proved that for any possible allocation of the data items that is not worse for any of the agents than opting out, there is a set of stable strategies (one for each server) which leads to this outcome. That is, suppose *alloc* is a specific allocation which all the servers prefer than opting out of the negotiation. A strategy for each of the servers can be designed such that the strategy profile will be an equilibrium. If the servers use this strategy profile, the negotiations will end at the first time period of the negotiation with the agreement *alloc*.

The details of the allocations that are not worse for any of the agents over opting out depend on the specific settings of the environment in a given negotiation session. Thus, there is no way to identify these allocations in advance. In addition, there are usually several allocations which are not worse for any of the agents than opting out. Finding all these allocations is intractable. In addition, after identifying these allocations the servers should agree upon one of them as the basis for the negotiation.[6] Of course, each of the servers may prefer a different allocation because it may yield a higher utility. A mechanism by which the servers can choose one of these profiles of stable strategies is presented. It leads to satisfactory results for all of the servers. In this mechanism each server proposes an allocation and the one which maximizes a social welfare criterion (e.g., the sum of the servers' utilities) is selected. Several heuristic search algorithms to be used by the servers to find such allocations were proposed.

There are situations where the servers have incomplete information about each other. For such situations a preliminary step was added to the strategic negotiation where the servers reveal some of their private information. When the servers use the revelation mechanism, it is beneficial for them to truthfully report their private information. After the preliminary step, the negotiation continues as in the complete information case and yields better results for all the servers than the static allocation policy currently used in EOSDIS. Thus, the overall process in this case is: First, each server broadcasts its private information. If a lie is detected, then the liar is punished by the group. In the next step each server searches for an allocation and then simultaneously each of them proposes one. The allocation which maximizes the pre-defined social-welfare criterion is selected. Then, the servers construct the equilibrium strategies based on the chosen allocation and they start the negotiation using the alternating offers protocol. In the first step of negotiations, the first agent proposes the selected allocation, and the others accept it.

In addition to the theoretical results, simulation results which demonstrate the effect of different parameters of the environment on the negotiation results are also presented. For example, when the servers are more willing to store data locally, better agreements can be reached. The reason for this is that in such situations there are less constraints on finding agreements which are better for all the servers than opting out, and it is easier to find a beneficial compromise. The servers are more willing to store data locally when the storage costs and the cost of delivery of documents stored locally to other servers is low.

In summary, the strategic negotiation model provides a unified solution to a wide range of problems. It is appropriate for dynamic real-world domains.

[6] In game-theory terminology, the game has multiple equilibria and the problem of the players is to convert into one of them.

In addition to the application of the strategic-negotiation model to data allocation problems in information servers, it was applied to resource allocation and task distribution problems, and the pollution allocation problem [38]. In all these domains the strategic-negotiation model provides the negotiators with ways to reach mutually beneficial agreements without delay. The application of the strategic-negotiation model to human high pressure crisis negotiations was also studied [113,41].

In the next section we will discuss using auctions, another game-theory based technique, for reaching agreements in multiagent environments.

3 Auctions for Resolving Conflicts

In many domains agreements should be reached by the agents concerning the distribution of a set of items. For example, in the information server environment, the agents need to decide on the allocation of datasets, i.e., the items under consideration are datasets. In resolving conflicts on the scheduling of the usage of a resource, an agreement should be reached on the time slots to be assigned to each agent. When the agents need to decide on task assignments, then the items are the tasks and a decision should be made on which agent will carry out a given task. Most of these conflicts can be resolved efficiently by providing the agents with a monetary system, modeling them as buyers and sellers, and resolving the conflicts using a money transfer [71]. For example, a server may "sell" a dataset to another server when relocating this dataset; a subcontractor may be paid in order to carry out a task.

Auctions have become an area of increased interest since a huge volume of economic transactions is conducted through these public sales. The formation of virtual electronic auction houses on the Internet [21] such as eBay [12] has even increased the interest in auctions.

There are two patterns of interactions in auctions. The most common are one-to-many auction protocols [79,1,17] where one agent initiates an auction and a number of other agents can bid in the auction, or many-to-many auction protocols [115] where several agents initiate an auction and several other agents can bid in the auction. Given the pattern of interaction, the first issue to determine is the type of protocols to use in the auction [34]. Given the protocol, the agents need to decide on their bidding strategy.

There are several types of one-to-many auctions which are used, including the English auction, first-price sealed-bid auction, second-price sealed-bid (Vickery auction), and the Dutch auction. The English auction is an ascending auction in which the price is successively raised until only one bidder remains, and that bidder wins the item at the finial price. In one variant of the English auction the auctioneer calls higher prices successively until only one willing bidder remains, and the number of active bidders is publicly known at all times. In other variants the bidders call out prices themselves, or have the bids submitted electronically and the best current bid is posted. The first-price sealed bid auction is a sealed-bid auction in which the buyer making the highest bid claims the object and pays the amount he has bid. The second-price auction is a sealed-bid auction in which the buyer making the highest bid claims the object, but pays only the amount of the second highest bid. In the Dutch auction, the auctioneer begins by naming a very high price and then lowers it continuously until some bidder stops the auction and claims the object for that price. In real world situations, each auction has its advantages and drawbacks [34,53]. In order to test various auction protocols in and to compare bidding strategies, a series of open-invitation events are conducted. These events are featuring software agents from all over the world competing in a market game. The agents need to bid to obtain travel packages for their clients [111]. Sandholm [83] surveys the existing auction protocols, and discusses certain known and new limitations of the protocol for multiagent systems, such as the possibility of bidder collusion and a lying auctioneer.

The Vickrey auction [107] is widely used in DAI [72,26,88,105,104] and in research on electronic commerce [105,104] for the case of one-to-many auctions. Under various assumptions, this protocol is incentive compatible, which means that each bidder has incentives to bid truthfully.

We demonstrate the application of the Vickery auction in the data-allocation problem.

Example 3. An auction protocol can be applied to the data-allocation problem discussed in examples 1 and 2 when a server is concerned with the data stored locally, but does not have preferences concerning the exact storage location of data stored in remote servers. For example, when each server provides information directly to a client which requires it, and obtains payments directly from this client.[7] According to this approach, the location of each data unit will be determined using an auction protocol, where the server bidding the highest price for obtaining the data will actually obtain it, but will pay the second-highest bid. This approach yields an efficient and fair solution [88], its implementation is simple, and the servers are motivated to offer prices which really reflect their utility.

There are situations in which the value of some items to a bidder depends upon which other items he or she wins. In such cases, bidders may want to submit bids for combinations of items. Such auctions are called *combinatorial auctions*. The main problem in combinatorial auctions is to determine the revenue maximizing set of non-conflicting bids. The general problem is NP-complete. Several researchers have been trying to develop polynomial algorithms, either for specific cases (e.g., [76]) or for finding sub-optimal solutions (e.g., [46,15,27].) Nisan [58] considers bidding languages and the allocation algorithm for combinatorial auctions.

[7] Note that the auction mechanism is not applicable in the environments that are considered in examples 1 and 2 where each server is concerned with the exact location of each dataset. In the auction mechanism if a server would like to store a dataset it can make a high bid, however, there is no way for a server to influence the location of datasets which are not stored locally.

Double auction is the most known auction protocol for many-to-many auctions. In a double auction, buyers and sellers are treated symmetrically with buyers submitting bids and sellers submitting minimal prices [114]. There are several algorithms used for matching buyers and sellers and for determining the transaction price. Preferably, the protocol will be *incentive compatible*, *individual rational* and Pareto optimal [115]. As mentioned above, an auction is incentive compatible if the agents optimize their expected utilities by bidding their true valuations of the goods. An auction is individual rational if participating in an auction does not make an agent worse off than not participating.

In the next section we will discuss an economic-based mechanism for distributed allocation which consists of auctions.

4 Market-Oriented Programming

Market-oriented programming is an approach to distributed computation based on market price mechanisms [109,112,16].

The idea of market-oriented programming is to exploit the institution of markets and models of them, and to build computational economies to solve particular problems of distributed resource allocation. This is inspired in part by economists' metaphors of market systems "computing" the activities of the agents involved. The modules, or agents, interact in a very restricted manner–by offering to buy or sell quantities of commodities at fixed unit prices. When this system reaches equilibrium, the computational market has indeed computed the allocation of resources throughout the system, and dictates the activities and consumptions of the various modules
(http://ai.eecs.umich.edu/people/wellman/MOP.html). Note that this approach does not necessarily require money transfer and it is applicable when there is incomplete information. However, it is applicable only when there are several units of each kind of goods and when the number of agents is large. Otherwise, it is not rational for the agents to ignore the effect of their behavior on the prices when they actually have an influence. Another issue is that there are situations in which reaching an equilibrium may be time consuming, and the system may not even converge [112]. It also requires some mechanism to manage the auctions, (possibly, a distributed mechanism, one for each type of goods.) A survey and a general discussion on the market-programming approach can be found in [112,110]. http://www2.elec.qmw.ac.uk/˜mikeg/text.html is a market based multi agent systems resource page.

5 Coalition Formation

Another important way for agents to cooperate is by creating coalitions [86,91,92]. The formation of coalitions for executing tasks is useful both in Multi-Agent Systems (MA) and Distributed Problem Solving (DPS) environments. However, in DPS, there is usually no need to motivate the individual agent to join a coalition. The agents can be built to try to maximize the overall performance of the system. Thus, only the problem of which coalitions should be formed (i.e., the structure of the coalitions) for maximizing the overall expected utility of the agents should be considered. However, finding the coalition structure that maximizes the overall utility of the system is NP-complete.

In Multi-Agent Systems (MA) of self-interested agents, an agent will join a coalition only if it gains more if it joins the coalition than it could gain previously. Thus, in addition to the issue of the coalition structure, the problem of the division of the coalition's joint utility is very important. Game theory techniques for coalition formation can be applied for solving this problem. Work in game theory such as [69,93,108,117] describes which coalitions will form in N-person games under different settings and how the players will distribute the benefits of the cooperation among themselves. This is done by applying several related stability notions such as the core, Shapley value and the kernel [30]. Each of the stability notions is motivated by a different method of measuring the relative strengths of the participating agents. However, the game-theory solutions to the coalition formation problem do not take into consideration the constraints of a multiagent environment, such as communication costs and limited computation time, and do not present algorithms for coalition formation.

The coalition formation of self-interested agents in order to satisfy goals is considered in [92]. Both the coalition structure and the division of the utility problems are handled. An anytime algorithm for forming coalitions that satisfy a certain stability based on the kernel stability criteria is developed. The properties of this algorithm were examined via simulations which showed the model increases the benefits of the agents within a reasonable time period, and more coalition formations provide more benefits to the agents. These results were applied to the formation of coalitions among information agents [35].

Sandholm et al. [80] focused on establishing the worst case bound on the coalition structure quality while only searching a small fraction of the coalition structures. They show that there is a minimal number of structures that should be searched in order to establish a bound. They present an anytime algorithm that establishes a tight bound within this minimal amount of search. If the algorithm is allowed to search further, it can establish a lower bound.

Sandholm and Lesser [84] developed a coalition formation model for bounded rational agents and present a general classification of coalition games. They concentrate on the problem of computing the value of a coalition and in their model this value depends on the computation time available to the agents.

Zlotkin and Rosenschein [118] study the problem of the utility division in Subadditive Task Oriented Domains that is a subset of the Task-Oriented Domains (see section 2.1). They consider only the grand coalition structure where all the agents belong to the same coalition and provide a linear algorithm that guarantees each agent an expected utility that is equal to its Shapley value. Ketchpel [33] presents a utility distribution mechanism designed to perform in similar situations where there is uncertainty in the utility that a coalition obtains.

Coalition formation in DPS environments in order to perform tasks is considered in [91]. In this case, only the coalition structure problem is considered. Efficient distributed algorithms with low ratio bounds and with low computational complexities are presented. Both agent coalition formation where each agent must be a member of only one coalition and overlapping coalitions are considered.

6 Contracting

An agent may try to contract out some of the tasks that it cannot perform by itself, or that may be performed more efficiently by other agents. One self-interested agent may convince another self-interested agent to help it with its task, by promises of rewards.

The main question in such a setting is how one agent can convince another agent to do something for it when the agents do not share a global task and the agents are self-interested. Furthermore, if the contractor-agent can choose different levels of effort when carrying out the task, how can the manager-agent convince the contractor-agent to carry out the task with the level of effort that the manager prefers without the need of the manager's close observation.

The issue of incentive contracting has been investigated in economics and game theory during the last three decades (e.g., [2,74,70,18,25,43]) These works in economics and game theory consider different types of contracts for different applications. Examples of these are contracts between a firm and an employer or employers (e.g., [55,4,5,48]); a government and taxpayers (e.g., [9]); a landlord and a tenant (e.g., [2]); an insurance company and a policy holder (e.g., [78,24,98,45]); a buyer and a seller (e.g., [50,54]); a government and firms (e.g., [51]); stockholders and managements (e.g., [2]); a professional and a client [90], etc. In these situations two parties usually exist. The first party (called "the agent" in economics literature) must choose an action or a level of effort from a number of possibilities, thereby affecting the outcome of both parties. The second party (named "the principal") has the additional function of prescribing payoff rules. Before the first party (i.e., the agent) chooses the action, the principal determines a rule (i.e., a contract) that specifies the fee to be paid to the other party as a function of the principal's observations. Despite the similarity of the above applications, they differ in several aspects, such as the amount of information that is available to the parties, the observations that are made by the principal, and the number of agents. Several concepts and techniques are applied to the principal-agent paradigm in the relevant economics and game theory literature.

A well-known framework for automated contracting is the Contract Net protocol. It was developed for DPS environments where all the agents work on the same goal. In the Contract Net protocol a contract is an explicit agreement between an agent that generates a task (the manager) and an agent that is willing to execute the task (the contractor). The manager is responsible for monitoring the execution of a task and processing the results of its execution, whereas the contractor is responsible for the actual execution of the task. The manager of a task announces the task's existence to other agents. Available agents (potential contractors) then evaluate the task announcements made by several managers and submit bids for the tasks they are suited to perform. Since all the agents have a common goal and are designed to help one another, there is no need to motivate an agent to bid for tasks or to do its best in executing them if its bid is chosen. The main problems addressed by [95,97,96] are task decomposition, sub-tasks distribution, and synthesis of the overall solution.

The Contract Net was used in various domains [65,60,49,89]. For example, a modified version of the Contract Net protocol for competitive agents in the transportation domain was presented in [79]. It provides a formalization of the bidding and the decision awarding processes, based on marginal cost calculations according to local agent criteria. More important, an agent will submit a bid for a set of delivery tasks only if the maximum price mentioned in the tasks' announcement is greater than what the deliveries will cost that agent. A simple motivation technique is presented to convince agents to make bids; the actual price of a contract is half way between the price mentioned in the task announcement and the bid price.

Contracting in various situations of automated agent environments is considered in [37] . These situations include certainty vs. uncertainty, full information vs. partial information, symmetric information vs. asymmetric information and bilateral situations vs. situations where there are more than two automated agents in the environment. For each of these situations appropriate economic mechanisms and techniques that can be used for contracting in automated agents environments are fitted from the game theory or economics literature. In all the situations that are considered, the agent that designs the contract is provided with techniques to maximize its personal expected utilities, given the constraints of the other agent(s).

Sandholm and his colleagues [81,82] developed a backtracking method called *leveled commitment contract* where each party can unilaterally decommit to a contract by paying a predetermined penalty. They show that such contracts improve expected social welfare even when the agents decommit strategically in Nash equilibrium.

7 Logical Approaches to Argumentation

Several researchers developed frameworks for negotiation through argumentation in which agents exchange proposals and counter-proposals backed by arguments that summarize the reasons why the proposal should be accepted. The argumentation is persuasive because the exchanges are able to alter the mental state of the agents involved.

Most of these framework are based on logical models of the mental states of the agents representing, for example, their beliefs, desires, intentions, and goals. The formal models are used in two manners. One use is as a specification for agent design [19]. In this role, the model constrains certain planning and negotiation processes. It can also be used to check the agents' behavior. Another use of the model is by the agents themselves.

Parsons and Jennings [64] drew upon a logic of argumentation to devise a system of argumentation and use it to implement a form of dialectic negotiation. In their context, an argument is a sequence of logical steps indicating support or doubt of a proposition. They have a function of flattening, which can measure the set of arguments into some metric of how favored the proposition is, by determining which class of acceptability the arguments belong to.

Qiu and Tambe [67] focus on negotiations between team members to resolve conflicts that arise from conflicting beliefs about different aspects of their environment, about resource availability, and about their own or teammates' capabilities and performance. The basis of such negotiations is inter-agent argumentation where agents assert their beliefs to others, with supporting arguments. Their approach is implemented in a system called CONSA (COllaborative Negotiation System based on Argumentation).

In [40] a logic is used in the above two ways. Using categories identified in human multi-agent negotiation, demonstrate how the logic can be used to specify argument formulation and evaluation. Furthermore, [40] presents a general Automated Negotiation Agent which was implemented, based on the logical model. Using this system, a user can analyze and explore different methods to negotiate and argue in a non-cooperative environment where no centralized mechanism for coordination exists. The development of negotiating agents in the framework of the Automated Negotiation Agent is illustrated with an example where the agents plan, act, and resolve conflicts via negotiation in a Blocks World environment.

The formal model of [40] consists of a set of agents, not necessarily cooperative, with the ability to exchange messages. Their mental states are characterized by using the notions of beliefs, goals, desires, intentions, and local preferences. Each agent has a set of desires. The agent's activities are motivated by the will to fulfill these desires. At any given time, an agent selects a consistent subset of its desires. This serves as its set of current goals. An agent ascribes different degrees of importance to different goals. It prefers to fulfill goals of higher importance. The set of goals motivate the agent's planning process.

The planning process may generate several intentions. Some of these are in what we would like to classify as the "intend-to-do" category and refer to actions that are within the direct control of the agent. Others are among the "intend-that" category [8,19,20,106]. These are propositions not directly within the agent's realm of control, that it must rely on other agents for satisfying.[8] Often, there is room for argumentation when intend-that actions are part of a plan. Argumentation is the means by which an agent, the persuader, attempts to modify the intention structure of another agent, the persuadee, to include

[8] The proposition may include a negation. When fulfillment of the proposition is beyond the control of the agent, it can be achieved by convincing another agent to abandon a relevant intention, or by convincing it to take an action that will make the proposition true.

the actions the persuader wants it to do. While an agent tries to influence the intentions of other agents, other agents may try to convince it as well. The role of persuader and persuadee is not fixed, but dynamically assumed during the agent interactions. Thus, during a negotiation process, each agent may update its intentions and goals after receiving a message from another agent. If the argumentation happens to fail, the agent which sent it must revise its arguments, its plans, and/or seek other sources of satisfying the portion of its plan in question.

An agent's belief set includes beliefs concerning the world and beliefs concerning mental states of other agents. An agent may be mistaken in both kinds of beliefs. It may update its beliefs by observing the world and after receiving messages from other agents. Each agent's actions are based upon its mental model of other agents.

Arguments serve either to add an intention to the persuadee's set or to retract an intention or to change the preferences of the persuadee. Below we present a list of several argument types. These argument types are not meant to constitute an exhaustive typology of arguments. Indeed, it has been pointed out [103] that it is not possible to present such an authoritative classification, since arguments must be interpreted and are effective within a particular context and domain. The six argument types that we present are ones that are commonly thought to have persuasive force in human negotiations [61,31,66]. Argumentations which were shown to be successful in human negotiation, may be also successful in automated agents' negotiations. Furthermore, we want our agents to be able to negotiate with humans, and therefore they need to be able to at least understand human argumentation. Moreover, the designers of the agents can follow the negotiation of the agents, if it is similar to human negotiation. The argument types we present are:

1. Threats to produce goal adoption or goal abandonment on the part of the persuadee.
2. Enticing the persuadee with a promise of a future reward.
3. Appeal to past reward.
4. Appeal to precedents as counterexamples to convey to the persuadee a contradiction between what she/he says and past actions.
5. Appeal to "prevailing practice" to convey to the persuadee that the proposed action will further his/her goals since it has furthered others' goals in the past.
6. Appeal to self-interest to convince a persuadee that taking this action will enable achievement of a high-importance goal.

Threats and promises are the most common arguments used in human negotiations [7]. An appeal to prevailing practice is the most common argument used in the legal system. Furthermore, it was found that presenting example instances (prevailing practice cases) is much more persuasive than presenting statistical summaries [36,101,59,23]. An "appeal to past promise" is supported by the cognitive dissonance theory [61] that assumes that a person seeks to maximize the internal psychological consistency of his/her cognition, and thus will be willing to keep his/her promises. This argument is also important in repeated interactions since agents prefer to maintain their credibility. The other two arguments, "an appeal to self interest" and "a counter example" are examples of arguments useful to persuade bounded rational agents which have limited inferential resources. More discussion on these arguments can be found in [40].

8 Conclusions

Game-theory and economics techniques seem to be very useful in the development of self-interested automated agents that act in a well-defined environment. Logical models provide a framework for argumentations. In this paper we emphasized formal techniques. We believe that using them in multi-agent systems is beneficial because there is a need to provide the agents with well-designed algorithms.

The choice of the specific technique for a given domain depends on the specification of the domain. For example, whether the agents are self interested, the number of agents in the environment, the type of agreement that they need to reach, and the amount and the type of information the agents have about each other.

While negotiation has been studied in other disciplines for many years, the study of negotiations of multi-agent environment is relatively new. In particular, there are many open questions with respect to applying formal models to multi-agent environments. The main challenge is how to maintain the useful results of the formal models, while adjusting them to real-world applications.

References

1. M. R. Andersson and T. W. Sandholm. Contract Types for Optimal Task Allocation: II Experimental Results. In *AAAI 1998 Spring Symposium: Satisficing Models*, Stanford University, California, 1998.
2. K. J. Arrow. The economics of agency. In J. Pratt and R. Zeckhauser, editors, *Principals and Agents: The Structure of Business*, pages 37–51. Harvard Business School Press, Cambridge, MA, 1985.
3. R. Azoulay-Schwartz and S. Kraus. Negotiation on data allocation in multi-agent environments. *Autonomous Agents and Multi-Agent Systems journal*, 2001. To appear.
4. S. Baiman and J. Demski. Economically optimal performance evaluation and control systems. *Journal of Accounting Research*, 18:184–220, 1980.
5. A. Banerjee and A. Beggs. Efficiency in hierarchies: implementing the first-best solution by sequential actions. *The Rand Journal of Economics*, 20(4):637–645, 1989.
6. M. Ben-Or and N. Linial. Collective coin flipping, robust voting games and minima of banzhaf values. In *Proc. 26th IEEE Symposium on the Foundations of Computer Science*, pages 408–416, Portland, 1985.
7. F. J. Boster and P. Mongeau. Fear-arousing persuasive messages. In R. N. Bostrom, editor, *Communication yearbook 8*. SAGE Publications, 1984.
8. M. E. Bratman. Shared cooperative activity. *The Philosophical Review*, 101:327–341, 1992.

9. B. Caillaud, R. Guesnerie, P. Rey, and J. Tirole. Government intervention in production and incentives theory: a review of recent contributions. *Rand Journal of Economics*, 19(1):1–26, 1988.
10. S.E. Conry, K. Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage negotiation for distributed satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Artificial Intelligence*, 21(6):1462–1477, December 1991.
11. D. Druckman. *Negotiations.* Sage Publications, London, 1977.
12. eBay. eBay - Your Personal Trading Community. http://www.ebay.com/aw/, 2001.
13. R. Fisher and W. Ury. *Getting To Yes: Negotiating Agreement Without Giving In.* Houghton Mifflin, Boston, 1981.
14. E. Freitsis. Negotiations in the pollution sharing problem. Master's thesis, Bar-Ilan University, 2000.
15. Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In Dean Thomas, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol1)*, pages 548–553, S.F., July 31–August 6 1999. Morgan Kaufmann Publishers.
16. C. Gerber, C. Russ, and G. Vierke. On the suitability of market-based mechanisms for telematics applications. In *Proc. of Agents99*, pages 408–409, 1999.
17. E. Gimenez-Funes, L. Godo, and J. A. Rodriguez-Aguilar. Designing bidding strategies for trading agents in electronic commerce. In *ICMAS98*, pages 136–143, Paris, 1998.
18. S. Grossman and O. Hart. An analysis of the principal-agent problem. *Econometrica*, 51(1):7–45, 1983.
19. B. J. Grosz and S. Kraus. Collaborative plans for complex group activities. *Artificial Intelligence Journal*, 86(2):269–357, 1996.
20. B. J. Grosz and S. Kraus. The evolution of sharedplans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*, pages 227–262. Kluwer Academic Publishers, 1999.
21. R. H. Guttman and P. Maes. Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In *The Second International Workshop on Cooperative Information agents (CIA98)*, pages 135–147, Paris, 1998.
22. Lavinia Hall, editor. *Negotiation: Strategies for Mutual Gain.* Sage, Beverly Hills, 1993.
23. R. Hamill, T. D. Wilson, and R. E. Nisbett. Insensitivity to sample bias: Generalizing from atypical cases. *Journal of Personality and Social Psychology*, 39:579–589, 1980.
24. M. Harris and A. Raviv. Some results on incentive contracts with applications to education and employment, health insurance, and law enforcement. *The American Economic Review*, 68(1):20–30, 1978.
25. J. Hirshleifer and J. Riley. *The Analytics of Uncertainty and Information.* Cambridge University Press, Cambridge, 1992.
26. B. Huberman and S. H. Clearwater. A multi-agent system for controlling building environments. In *Proceedings of ICMAS-95*, pages 171–176, San Fransisco, 1995.
27. L. Hunsberger and B. Grosz. A combinatorial auction for collaborative planning. In *ICMAS-2000*, Boston, USA, 2000.
28. N. R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated haggling: Building artificial negotiators. In *Pacific Rim International Conference on Artificial Intelligence*, page 1, 2000.

29. R. Johnson. *Negotiation basics.* Sage, Newbury Park, 1993.
30. J. P. Kahan and A. Rapoport. *Theories of coalition formation.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.
31. M. Karlins and H. I. Abelson. *Persuasion: How opinions and attitudes are changed.* Springer Publishing Company, Inc., second edition, 1970.
32. C.L Karrass. *The Negotiating Game: How to Get What You Want.* Thomas Crowell Company, NY, 1970.
33. S. P. Ketchpel. Forming coalitions in the face of uncertain rewards. In *Proc. of AAAI94*, pages 414–419, Seattle, Washington, 1994.
34. P. Klemperer. Auction theory: A guide to literature. *Journal of Economic Surveys*, 13(3), 1999.
35. M. Klusch and O. Shehory. A polynomial kernel-oriented coalition formation algorithm for rational information agents. In *Proc. of ICMAS-96*, pages 157–164, Kyoto, Japan, 1996.
36. R. R. Jr. Koballa. Persuading teachers to reexamine the innovative elementary science programs of yesterday: The effect of anecdotal versus data-summary communications. *Journal of Research in Science Teaching*, 23:437–449, 1986.
37. S. Kraus. An overview of incentive contracting. *Artificial Intelligence Journal*, 83(2):297–346, 1996.
38. S. Kraus. *Strategic Negotiation in Multiagent Environments.* MIT Press, Cambridge, USA, 2001.
39. S. Kraus and D. Lehmann. Designing and building a negotiating automated agent. *Computational Intelligence*, 11(1):132–171, 1995.
40. S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1-2):1–69, 1998.
41. S. Kraus and J. Wilkenfeld. A strategic negotiations model with applications to an international crisis. *IEEE Transaction on Systems Man and Cybernetics*, 23(1):313—323, 1993.
42. D. Kreps and R. Wilson. Sequential equilibria. *Econometrica*, 50:863–894, 1982.
43. J. Laffont and J. Tirole. *A Theory Of Incentives in Procurement and Regulation.* The MIT Press, Cambridge, Massachusetts, 1993.
44. Susan E. Lander and Victor R. Lesser. Customizing distributed search among agents with heterogeneous knowledge. In *Proc. first int. conf. on Information Knowledge Management*, pages 335–344, Baltimore, 1992.
45. M. Landsberger and I. Meilijson. Monopoly insurance under adverse selection when agents differ in risk aversion. *Journal of Economic Theory*, 63:392–407, 1994.
46. Daniel Lehmann, Liadan Ita O‘Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC‘99)*, pages 96–102, N.Y., November 3–5 1999. ACM Press.
47. R. D. Luce and H. Raiffa. *Games and Decisions.* John Wiley and Sons, New York, 1957.
48. I. Macho-Stadler and J. Pérez-Castrillo. Moral hazard and cooperation. *Economics Letters*, 35:17–20, 1991.
49. T. W. Malone, R. E. Fikes, K. R. Grant, and M. T. Howard. Enterprise: A market-like task schedule for distributed computing environments. In B. A. Huberman, editor, *The Ecology of Computation*, pages 177–205. North Holland, 1988.
50. S. Matthews. Selling to risk averse buyers with unobservable tastes. *Journal of Economy Theory*, 30:370–400, 1983.

51. R. P. McAfee and J. McMillan. Bidding for contracts: a principal-agent analysis. *The Rand Journal of Economics*, 17(3):326–338, 1986.
52. T. Moehlman, V. Lesser, and B. Buteau. Decentralized negotiation: An approach to the distributed planning problem. *Group Decision and Negotiation*, 2:161–191, 1992.
53. D. Monderer and M. Te nnenholtz. Optimal auctions revisited. *Artificial Intelligence Journal*, 120:29–42, 2000.
54. R. Myerson. Mechanism design by an informed principal. *Econometrica*, 51:1767—1798, 1983.
55. B. Nalebuff and J. Stiglitz. Information, competition, and markets. *American Economic Review*, 73(2):278–283, 1983.
56. NASA. EOSDIS Home Page.
http://www-v0ims.gsfc.nasa.gov/v0ims/index.html, 1996.
57. J. F. Nash. Two-person cooperative games. *Econometrica*, 21:128–140, 1953.
58. Noam Nisan. Bidding and allocation in combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 1–12, 2000.
59. R. E. Nisbett, E. Borgida, R. Crandall, and H. Reed. Popular induction: information is not necessarily informative. In J. S. Carroll and J. W. Payne, editors, *Cognition and social behavior*. Lawrence Erlbaum, 1976.
60. T. Ohko, K. Hiraki, and Y. Anzai. Reducing communication load on contract net by case-based reasoning:extension with directed contract and forgetting. In *Proceedings of the First International Conference on Multi–Agent Systems*, Cambridge, MA, 1995. MIT Press.
61. Daniel J. O'Keefe. *Persuasion: Theory and Research*. SAGE Publications, 1990.
62. M. J. Osborne and A. Rubinstein. *Bargaining and Markets*. Academic Press Inc., San Diego, California, 1990.
63. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, Massachusetts, 1994.
64. S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.
65. H. van Dyke Parunak. Manufacturing experience with the contract net. In *Proceedings of the 1995 Distributed Artificial Intelligence Workshop*, pages 67–91, December 1995.
66. D. G. Pruitt. *Negotiation Behavior*. Academic Press, New York, N.Y., 1981.
67. Z. Qiu and M. Tambe. Flexible negotiations in teamwork: Extended abstract. In *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*, 1998.
68. H. Raiffa. *The Art and Science of Negotiation*. Harvard University Press, Cambridge, MA, 1982.
69. A. Rapoport. *N-Person Game Theory*. University of Michigan, Michigan, 1970.
70. E. Rasmusen. *Games and Information*. Basil Blackwell Ltd., Cambridge, MA, 1989.
71. Amir Ronen. Algorithms for rational agents. In *Conference on Current Trends in Theory and Practice of Informatics*, pages 56–70, 2000.
72. J. S. Rosenschein. *Rational Interaction: Cooperation Among Intelligent Agents*. PhD thesis, Stanford University, 1986.
73. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Boston, 1994.
74. S. Ross. The economic theory of agency: The principal's problem. *The American Economic Review*, 63(2):134–139, 1973.

75. A. E. Roth. *Axiomatic Models of Bargaining.* Lecture Notes in Economics and Mathematical Systems No. 170. Springer-Verlag, Berlin, 1979.
76. M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. Technical Report 95-09, DIMACS, April 19 1995.
77. A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982.
78. A. Rubinstein and M. Yaari. Repeated insurance contracts and moral hazard. *Journal of Economic Theory*, 30:74–97, 1983.
79. T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of AAAI-93*, pages 256–262, Washington D.C., July 1993.
80. T. Sandholm, K. Larson, M. R. Andersson, O. Shehory, and F. Tohm. Coalition structure generation with worst case guarantees. *Artificial Intelligence Journal*, 111:209–238, 1999.
81. T. Sandholm, S. Sikka, and S. Norden. Algorithms for optimizing leveled commitment contracts. In *Proc. of IJCAI99*, pages 535–540, Stockholm, Sweden, 1999.
82. T. Sandholm and Y. Zhou. Surplus equivalence of leveled commitment contracts. In *Prof. of ICMAS-2000*, pages 247–254, Boston, MA, 2000. IEEE Computer Society.
83. T. W. Sandholm. Limitations of the vickrey auction in computational multiagent systems. In *International Conference on Multiagent Systems (ICMAS-96)*, pages 299–306, Kyoto, Japan, 1996.
84. T. W. Sandholm and V. R. Lesser. Coalition formation among bounded rational agents. In *Proc. of IJCAI-95*, pages 662–669, Montrèal, 1995.
85. T. W. Sandholm and V. R. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *First International Conference on Multiagent Systems (ICMAS-95)*, pages 328–335, San Fransisco, 1995.
86. T. W. Sandholm and V. R. Lesser. Coalition formation among bounded rational agents. *Artificial Intelligence*, 94(1-2):99–137, 1997. Special issue on Principles of Multi-Agent Systems.
87. R. Schwartz and S. Kraus. Negotiation on data allocation in multi-agent environments. In *Proc. of AAAI-97*, pages 29–35, Providence, Rhode Island, 1997.
88. R. Schwartz and S. Kraus. Bidding mechanisms for data allocation in multi-agent environments. In Munindar P. Singh, Anand S. Rao, and Michael J. Wooldridge, editors, *Intelligent Agents IV: Agent Theories, Architectures, and Languages*, pages 61–75. Springer-Verlag, 1998.
89. Sandip Sen and Edmund Durfee. A contracting model for flexible distributed scheduling. *Annals of Operations Research*, 65:195–222, 1996.
90. S. Shavell. Risk sharing and incentives in the principal and agent relationship. *Bell Journal of Economics*, 10:55–79, 1979.
91. O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligentce*, 15(3):218–251, 1998.
92. O. Shehory and S. Kraus. Feasible formation of stable coalitions among autonomous agents in non-super-additive environments. *Computational Intelligence*, 15(3):218–251, 1999.
93. K. I. Shimomura. The bargaining set and coalition formation. Technical Report 95-11, Brown University, Dept. of Economics, 1995.

94. C. Sierra, P. Faratin, and N. Jennings. A service-oriented negotiation model between autonomous agents. In *Proc. 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97)*, pages 17–35, Ronneby, Sweden, 1997.
95. R. Smith and R. Davis. Framework for cooperation in distributed problem solvers. *IEEE Transactions on Systems, Man and Cybernetic*, C-29(12):61–70, 1981.
96. R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem soler. *IEEE Transactions on Computers*, 29:1104–1113, 1980.
97. R.G. Smith and R. Davis. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.
98. M. Spence and R. Zeckhauser. Insurance, information and individual action. *The American Economic Review*, 61(1):380–391, 1971.
99. K. P. Sycara. *Resolving Adversarial Conflicts: An Approach to Integrating Case-Based and Analytic Methods.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, 1987.
100. K. P. Sycara. Persuasive argumentation in negotiation. *Theory and Decision*, 28:203–242, 1990.
101. S. E. Taylor and S. C. Thompsons. Staking the elusive vividness effect. *Psychological Review*, 89:155–181, 1982.
102. Jean Tirole. *The Theory of Industrial Organization.* The MIT Press, Cambridge, MA, 1988.
103. S. Toulmin, R. Rieke, and A. Janik. *An introduction to reasoning.* Macmillan Publishing Co., Inc., 1979.
104. M. Tsvetovatyy, M. Gini, B. Mobasher, and Z. Wieckowski. Magma: An agent-based virtual market for electronic commerce. *Applied Artificial Intelligence, special issue on Intelligent Agents*, 6:501–523, 1997.
105. M. B. Tsvetovatyy and M. Gini. Toward a virtual marketplace: Architectures and strategis. In *The first International Conference on the Practical Application of Intelligent Agents and Multi Agents Technology*, pages 597–614, London, 1996.
106. B. Vermazen. Objects of intention. *Philosophical Studies*, 71:223–265, 1993.
107. William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J. of Finance*, 16:8–37, March 1961.
108. R. Vohra. Coalitional non-cooperative approaches to cooperation. Technical Report 95-6, Brown University, Dept. of Economics, 1995.
109. M. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.
110. M. Wellman. Market-oriented programming: Some early lessons. In S. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific, Singapore, 1996.
111. M. Wellman and M. Wurman. TAC: Trading agent competition. http://auction2.eecs.umich.edu/game.html, 2001.
112. M. P. Wellman and P.R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125, 1998.
113. J. Wilkenfeld, S. Kraus, K. Holley, and M. Harris. Genie: A decision support system for crisis negotiations. *Decision Support Systems*, 14:369–391, 1995.
114. R. Wilson. Incentive efficiency of double auctions. *Econometrica*, 53(5):1101–1116, 1985.

115. P. R. Wurman, W. E. Walsh, and M. P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24:17–27, 1998.
116. Dajun Zeng and Katia Sycara. Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, 48:125–141, 1998.
117. L. Zhou. A new bargaining set of an n-person game and endogenous coalition formation. *Games and Economic Behavior*, 6:512–526, 1994.
118. G. Zlotkin and J. S. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proc. of AAAI94*, pages 432–437, Seattle, Washington, 1994.

Agents' Advanced Features for Negotiation and Coordination

Eugénio Oliveira

Faculdade de Engenharia, Universidade do Porto, NIAD&R-LIACC
Rua Dr. Roberto Frias 4200-465 Porto, Portugal
eco@fe.up.pt

Abstract. Agent-based systems suitable for dealing with applications where the environment is both dynamic and populated with competitors demand for sophisticated characteristics including adaptation, negotiation and co-ordination. We here briefly summarize some proposals on agents' negotiation capabilities including adaptation through reinforcement learning as well as qualitative multi-criteria negotiation and coalition formation protocols. Also, and inspired by robosoccer domain, some basic hints on knowledge representation for agents' team work are here described. All those proposals on automatic negotiation have been implemented through agent-based systems for different application domains (MACIV, SMACE, ForEV).

1. Introduction

The main objective of this paper is to present some recent developments we have done in our group (Distributed AI Group at the University of Porto) on possibly relevant aspects of agents' capabilities for interaction.

Different authors are proposing new improvements to agents characteristics, making it more meaningful to talk about agents' autonomy, pro-activity, mutual coordination and high level cognition through so-called "mental attitudes" [26, 3]. It is also fair to quote here the excellent and pioneering work done on Agents' negotiation reported in [16].

It is our intention here, to elaborate upon contributions on two of those agents' most important features: autonomy and mutual coordination.

One way of enhancing agent's autonomy in dynamic environments, is to endow its architecture with learning capabilities. Agent learning process may include several different facets, from simple, step by step, adaptation to the changes in a dynamic environment, to heavier and more sophisticated processes of gathering new knowledge (about the environment and the other agents) based on the past history.

In addressing the agent learning feature in a multi-agent environment, our motivation was not only to build up single-agent learning mechanisms that, although situated, do not take into account other agents' presence in the environment. On the contrary, we have been interested in learning mechanisms that make agents learn mostly from the intentional interaction with others, while pursuing their own specific goal.

Agents we are dealing with, evolve in an environment that is only partially accessible and known and, most important, exhibiting dynamic characteristics at different levels and degrees. Trading with counter-parts in the presence of competitors, negotiating possible future collaboration together with potential partners or performing together with a team of agents competing against others in an adversarial environment, all these situations imply the beneficial of introducing some agents' learning capabilities. Moreover, they all point out for the importance of methods relying on the extraction of appropriate feedback from the environment.

Typically, learning algorithms using feedback as a trigger for the learning process, can be distinguished depending on situations where the desired action is known from the beginning, and those where the feedback value it is only related with a measure of utility that has to be maximised. The former case, where the environment may be seen as a kind of teacher, encompasses the supervised learning class of algorithms, while the latter, where the environment may be seen as displaying the role of a critic, relates to reinforcement learning algorithms.

Our main effort on this particular subject has been in adapting reinforcement learning algorithms for the sake of improving agents' performance, through interaction in a dynamic environment populated with other agents.

Following Weiss and Sen [17], we would better speak of agents' adaptation instead of learning once the aim is to induce incremental self-modifications enabling agents to "survive" (to get deals, to win competitions…) in changing environments.

Adaptive agents can be seen as having a high degree of autonomy and rationality once their decision-making capability become more directly connected with their own specific environment related experience. Another decision-making key point is related to bidding under multi-issue negotiation (we propose the use of a qualitative feedback) as well as combining different tactics for determining appropriate agents' behaviors according to the current scenario. All these contributions had in mind to address different facets of multi-agent systems suitable for the application domains we have been involved with, like distributed resources management (MACIV system [11]) and electronic commerce (SMACE system [1]).

A different aspect we also briefly address below, is agents' coordination in an adversarial environment. We have chosen simulated "robosoccer" competition as highly demanding scenario simultaneously including both aspects requiring agents' coordination policies: Collaboration for joint work, and competition between different teams in the same scenario.

Moreover, coordination here has to be achieved in a permanently changing environment and, therefore, planned joint activities are all the time being challenged by other possibly more urgent reactions. This phenomenon may lead to an unexpected, undesirable and serious, lack of agents' coordination. Our approach also takes this problem into consideration and proposes a solution.

This paper is organised as follows: In this first section, we gave an overview of the subjects we are currently investigating together with our motivation. Next section will introduce our perspective on agents' autonomy together with our contribution to self-adaptation and agents' negotiation capabilities. Section 3 is devoted to some hints on agents' coordination, mainly in so-called semi-adversarial environments. Final section is a short conclusion.

2. Autonomy for Agents

New computer-based applications with increasingly higher complexity, running in dynamic distributed environments require, systems with better decision-making capabilities in order to intelligently satisfy their own objectives. Speaking of agents, that are good candidates for the design and implementation of such systems, we may say that they should include a high degree of autonomy in distributed, dynamic and often open environments.

Cognitive Agents

Agent's definition is here considered according to [27]. The same authors also define intelligent agents in terms of flexible autonomous actions, where flexibility encompasses reactivity after perceiving the environment, pro-activity by taking the initiative and social ability through intelligent interactions with others in order to satisfy their own objectives.

Rational or Cognitive agents are those that use a formalisation as internal specification language for determining the way they reason, eventually leading to specific actions in the environment. Logic of different kinds (modal, temporal, intentional, non-monotonic) have been traditional candidates for describing agents' reasoning process, which is mainly based on cognitive-like concepts. High level cognitive primitives shall involve precise and unambiguous definitions of so called "mentalistic" concepts as knowledge, intentions, desires or believes. It has been argued that "specifications derived from cognitive notions are perhaps the most significant of the AI contribution to agents" [19]. Agents' reasoning process can then follow from the logic-based interrelationships between those "mentalistic" concepts, as it is the case in the BDI architecture.

Besides using model-checking for Agent's specification instead of logic theorem proving, a further simplification on cognitive agents architectures has been introduced by M.Georgeff et al., (see for example [3]) that, although constraining some of the expressive power of the logic-based representation formalism, provides a more practical agent's reasoning system. Those above mentioned authors coined such system as PRS (Procedural Reasoning System).

Agents, that are conscious of their own goals, may use that practical reasoning ability in order to decide on the best action to perform in the environment at any specific situation. However, through their practical reasoning capabilities, agents still have to reason about two different aspects: Deliberating about the selection of what goals to pursue, and deciding on how to achieve them. These two aspects are not completely separated, once the latter, which involves reasoning about planing future actions, and being driven be intentions may also interfere with - putting additional constraints to - future deliberations.

Negotiation

Automated negotiation between self-interested agents is of increasing importance in distributed and rational Multi-agent systems [15, 16]. Contrary to the Distributed Problem Solving perspective, where it is the system designer who imposes both an interaction protocol and a co-operative procedure, in Multi-agent systems, the agents, although provided with a protocol for interaction, they will choose their own local reasoning and decision making policies.

The goal of the negotiation process is maximisation of the utility of a future decision. Agents get involved in a negotiation process if it is individually rational, that is, if the agent's payoff in the negotiated solution is no less than the payoff that the agent would get by not participating in the negotiation [15].

Negotiation Model

In an Electronic Market environment, each agent has an objective that specifies its intention to buy or sell a specific product. That objective has to be achieved in a certain amount of time, specified by a deadline. Negotiation stops when this deadline is reached.

The negotiation model that we have adopted is based on the one in [2]. Therefore, we are mainly concerned about multilateral negotiations over a set of issues. Multilateral refers to the ability of buyer and seller agents to manage multiple simultaneous bilateral negotiations with other seller or buyer agents. In auction terms, it relates to a *sealed-bid continuous double auction*, where both buyers and sellers submit bids (proposals) simultaneously and trading does not stop as each auction is concluded (as each deal is made). Negotiation is achieved through the exchange of proposals between agents. The negotiation can be made over a set of issues, instead of the single-issue price found in most auctions. A proposal consists of a value for each of those issues and is autonomously generated by the agent's strategy.

The proposal evaluation is based on *Multi-Attribute Utility Theory (MAUT)*.

Strategic Reasoning

We here distinguish between more simple and direct agent's tactical reasoning and a more sophisticated and dynamic strategic reasoning.

A tactic is a linear combination of functions that generates a value for a simple negotiation issue [9]. We borrowed from [2, 9] several different possible tactics for agents that are dependent either from time, resources availability or opponents' behaviour.

Whereas time-dependent tactics depend on a predictable factor, it is difficult to foresee the results of applying resource, or behaviour, dependent ones, since they depend on "run-time variations" of several different factors.

Adaptive agents may use combinations of tactics as the underlying philosophy of implementing their strategy. Tactics can be combined using different weights, representing the relative importance of each criterion in the overall strategy. The multi-issue's values that will compose the proposal will be calculated by weighting accordingly to the values proposed by each one of the tactics used.

However, in repeated negotiations, agents should be capable of taking advantage of their own experience. This consideration led us [1] to the development of an agent that, enhanced with learning capabilities, can increase its performance as it runs more and more negotiations. We have called it the *AdaptiveBehaviourAgent (ABA)*. Tactics provide a way of adaptation, in some degree, to different situations, considering certain criteria. But it is not clear what tactics should be used in what situations. The ultimate goal of our adaptive agent is to learn just that.

The idea is to define a S*trategy* as the way in which the agent changes the tactic combination over time. In order to do that, we have used a kind of automated learning that can take place online, from the interaction with the environment: Reinforcement Learning [22]. It is also the most appropriate learning paradigm for dynamic environments, such as the one we are addressing.

By applying this kind of learning process, we aimed at enhancing those adaptive agents with the ability for winning deals in the presence of competitors as well as increasing the utility of those deals. We intended to check if the agents adapt to a given market environment, associated with the transaction of a given type of product.

In dynamic environments, such as an electronic market, actions are *non-deterministic*, in the sense that they do not always lead to the same results, when executed in the same state. For this reason, we implemented a specific kind of Reinforcement Learning – *Q*-learning – that estimates the value of executing each action in each state (also known as the quality *Q*). In our environment, actions are the result of weighted combinations of tactics that will be used in the proposal generation process. The characterisation of the states is a major factor to the success of the algorithm implementation, and will determine the relevance of the results obtained. In our case, we considered two main variables: the *number of negotiating agents* and the *time available for negotiation*.

Updating the *Q* values associated with each action-state pair – *Q(s,a)* – consists of rewarding those actions that leaded to good results while penalising the ones that failed to achieve the agent's goal. The general *Q*-learning update formula is the following:

$$Q(s,a) = Q(s,a) + \alpha\,[r + \gamma\, max_{a'} Q(s',a') - Q(s,a)]$$

where α is the learning rate, representing the impact of the update in the current value; *r* is the reward obtained by executing action *a* in state *s*; γ is the discount factor, meaning the importance of future *Q* values (in future states) to the *Q* currently being updated; $max_{a'} Q(s',a')$ is the maximum *Q* value for the actions in the next state.

For the *ABA* agents, actions leading to deals are rewarded with a function depending on the deal values' utility and on the average utility obtained so far. This allows us to distinguish, among the deals obtained, those that correspond to higher utilities. Considering the average utility takes into account, when classifying the goodness of a deal, the characteristics of the environment (the difficulties) that the agent is repeatedly facing; the same deal in harder conditions should have a greater reward because it is closer to the best possible deal. Goal failure imposes penalisation (negative reward) to the last action used.

Action selection is another important aspect of the Reinforcement Learning paradigm. The simplest rule would be to select the action with the biggest *Q* value. Yet, this rule does not consider that there may exist non-selected actions that may perform better. Furthermore, in dynamic environments and therefore *non-deterministic*, an action does not always lead to the same results. In fact, to obtain a continued reward of great value, the agent should prefer actions that were considered good in the past, but in order to find them it must try actions that were never selected before. This dilemma leads us to the need of a compromise between *exploitation* (to take advantage of good quality actions) and *exploration* (to explore unknown actions or those with less quality). To satisfy this compromise, *Softmax* policy has been selected. It uses a degree of exploration τ (the temperature) for choosing between all possible actions, while considering their ranking.

In order to make it possible for the agent to increase the utility obtained with the deals made, it is necessary that the agent does not prefer the first action leading to a deal. Before the agent tries enough actions, it has got an incomplete knowledge of the environment, that is, it might know what action to use to likely get a deal (because unsuccessful actions are penalised), but not what the best actions are (those that result in higher utilities). To enforce the agent to try all the actions available before preferring the best ones, we enforce *optimistic initial values*. This means that all the Q values associated with the actions are initialised to a value greater than the expected reward for them. This measure increases, independently of the action selection parameters chosen, the initial action exploration, since the Q values will then be updated to more realistic lower values.

Advanced Negotiation

Most of the well-known negotiation protocols deal with a single dimension (usually the price). Moreover, Wellman [25] also says that the preferences' reduction to the price single figure is what characterises the market. However, this seems to be a bit simplistic and unrealistic for several application domains like B2B interaction.

Here we are most concerned with the use of Software Agents for a rich (multi-dimensional) and flexible (adaptive) design of the Negotiation phase protocol.

Some authors [24] [10] advocate the use of auction protocols for agent-mediated Electronic Commerce, arguing that they are widely recognised by Economists as the most efficient way of resolving one-to-many negotiations.

Other authors point out the limitations of auctions' protocols and look for more flexible negotiation models [5]. They stress out that online auctions are in fact less efficient and more hostile than it would be desired. For example, the winner's curse, i.e. "the winner pays more than the real value of the product", seems to be a consequence of auctions and, therefore, they propose more co-operative multi-attribute decision analysis tools and negotiation protocols using distributed constraint satisfaction policies to support it.

However, both lines of research are aware of the need of enhancing simple price-based auction mechanisms, transforming them into new protocols encompassing multi-dimensional issues to be negotiated among the market participants.

This kind of negotiation is no longer of a win-lose type (zero-sum game on the game theory taxonomy) and becomes one of the win-win type (non-zero sum game) allowing agents to co-operate when negotiating over multiple dimensions. Therefore, co-operative negotiations can be described as the decision making process of resolving a conflict involving two or more parties over multiple independent, but non-mutually exclusive goals [8].

Issues on Multi-criteria Negotiation

It is not crucial whether we use multi-lateral negotiations or auction protocols for agent-mediated negotiation as far as we enhance the negotiation protocol capabilities with multi-attribute bids and appropriate evaluation functions.

Multi-criteria negotiation is extremely useful in a large number of situations. Situations where more than one single parameter is simultaneously taking part in the same negotiation are common in real commercial scenarios where buyers are confronted with several different appealing proposals.

In single-issue negotiation the value of a parameter is negotiated in such a way that the different negotiators try to defeat their opponents by offering a better proposal. The preferences of the buyer can be either implicit (the lower is the price the better for the buyer) or explicit with the buyer publicising its preference.

In one of our application domains, we are using a platform - ForEV- as an Electronic Market for Virtual Enterprise formation [12]. Here Agents representing *Organisations* exchange bids through a *Market Agent*. A state is defined by the n-uple:

In our application domain, a state is defined by the n-uple:

$s = (V_1, V_2, \dots V_n)$ V_i is the value for attribute i

The state will also represent the bid sent to the *Market Agent*.

We define an action by the n-uple:

$a = (A_1, A_2, \dots, A_n)$, $A_i \in$ {increase, decrease, maintain}

Continuous attributes' values can be increased (decreased) by pre-defined amount steps. Discrete attributes' values can be increased (decreased) by moving to the next (previous) element in the enumerated domain.

When an *Organisation Agent* receives a feedback message from the *Market Agent* to its previous bid (this bid represents the state s that results from performing action a_p in state s_p), it tries to formulate a new bid following the procedure:

1. Calculate a reward value for the previous bid (that also defines the current state s). This reward value is calculated using the qualitative evaluation included in the received feedback message:
2. $$r = n \quad \text{, if winner}$$ $$= n - \sum_i penalty^i \quad \text{, if not winner} \qquad 0 \le penalty \le 1$$
3. In the above formula, penalty is a parameter that decreases the reward value for bids with low evaluated values (bids are evaluated by the *Market Agent*). This parameter should be defined by each negotiating agent according to its own criteria and its value increases with the evaluations' category: low (high), very_low (very_high), extremely_low (extremely_high). In this formula, attributes should appear in the same order they apper in the feedback message sent by the *Market Agent* (in this feedback message, attributes are ordered by *Market Agent*'s order of importance).
4. Update the pair state/action Q-value ($Q(s_p,a_p)$) using the Q-learning formula presented above.
5. Store the current sate s, and its specific associated reward value r.
6. Derive all **new** promising actions a' taking into account that specific feedback message. For instance if the feedback message evaluates *attribute_2* as *too_low*, one possible promising action is to increase *attribute_2* and maintain all the others.

7. Store pairs *s/a'* with a default Q-value ($Q(s,a') = default$).
8. Move to the state $s*$ with maximum reward value.
9. Choose the action $a*$ with higher probability value, according to the Boltzmann exploration formula, (note that actions not ever tried are stored in that state $s*$ with Q-value=*default*).
10. Perform action $a*$ and send the new bid (state) out to the *Market Agent.*

Bid Evaluation

In order to select the most promising partners in a Virtual Enterprise formation scenario [12] (or, in a more general electronic market environment, the most favourable buyers/sellers for a business transaction) a negotiation process is started by the *Market Agent.* This negotiation process comprises several rounds, starting when the *Market Agent* sends an announcement for all agents registered in the electronic market. The negotiation ends when a deadline is reached or a satisfactory proposal is received. At each negotiation round bids are evaluated, and the one with the higher evaluation value is considered the winner in current round. Received bids comprise a solution to a specific needed goal, and include proposed values for multiple issues.

Bid evaluation is done by means of a multi-criteria function that encodes the attributes' and attributes values' preferences initially stipulated by the *Market Agent.* This multi-criteria evaluation function is defined by the following formula:

$$Ev = \frac{1}{Deviation}$$

$$Deviation = \frac{1}{n} * \sum_{i=1}^{n} \frac{i}{n} dif(PrefV_i, V_i)$$

where n = number of attributes for a specific goal.

In the formula above, each parcel should be presented in increasing order of preference. Attribute identified by number 1 (*i=1*) is the one that *Market Agent* classifies as the least important, and attribute identified by number n (*i=n*) is the most important.

The function $dif(PrefV_i, V_i)$ quantifies for an attribute *i*, the degree of acceptability of the current value (V_i) proposed by a specific organisation agent when compared to its preferable value *(PrefVi)* for the *Market Agent.* If the attribute values domain is of continuous values, this quantification is simply a normalised difference between the two values V_i and $PrefV_i$. If the attribute values domain is of discrete values, the result of the $dif(PrefV_i, V_i)$ function is now calculated as a normalised difference between the preference attached to V_i and $PrefV_I$, which can be calculated as the difference between the relative position of the two values in the enumerate domain values specification known to the *Market Agent* .

The winner bid in the current round is selected as the one that presents the highest evaluation value, since it is the solution that contains attributes values closer to the preferable values. The winner bid in this current round is compared with the winner bid in all past rounds, and the best one is selected.

All other bids are compared to the winner bid in order to inform the respective agents about the reason why their bids were not selected. A comparison is made for the values proposed for the most important attributes (from *Market Agent* point of view, of course). These values are then classified in a qualitative way: low (high), very_low (very_high), extremely_low (extremely_high).

Let us suppose that δ_i (i=1,2) splits the domain into the three mentioned regions, and *w* is the winner bid.

```
For all bids $k \neq w$, do
    For all the most important attributes $i$, do
        $\Delta_{i,k} = \text{dif}(V_{i,k}, V_{i,w})$
        If $\Delta_{i,k} < \delta_1$ then **Evaluation** = low (high)
        Else If $\Delta_{i,k} < \delta_2$ then **Evaluation** = very_low (very_high)
        Else **Evaluation** = extremely_low (extremely_high)
```

This information is included in the message that is sent out to all bidding agents except the currently winning agent, as a reason why their bids have not been selected.

Such information is a list where each element is a pair (At_{Id}, Evaluation), where At_{Id} means the attribute identification. Agents will receive this information and will formulate new bids to send to *Market Agent* in the next negotiation round.

The negotiation process ends when:

- Market Agent receives a bid that has a satisfactory evaluation value (Ev). This bid is the winner bid.
- A deadline is reached. The winner bid is the one that presents the highest evaluation value among all bids received until then.

Agents Coalition Formation

In many realistic situations agents may benefit by finding out a co-operative agreement enabling them to appear together as a single entity facing other agents.

How can situations where several agents in a coalition can compete with other agents or coalitions of agents be handled?

The negotiation algorithm must now have the capabilities to allow a different and appropriate bid generation process [11].

An appropriate negotiation algorithm has to consider different steps as follows:

- Selection phase – An announcer collects the bids, and rejects those coming from agents that do not fulfil (from its point of view) basic pre-conditions to be eligible. However, the announcer must now derive the possible coalitions that can solve the task in hand. This is an exponentially complex problem because with N agents involved we will have 2^N-1 possible coalitions. However, a large number of these possibilities can be immediately rejected through the application of some simple constraints. In each one of the approved coalitions one of its members has then to be designated as “co-ordinator”. The co-ordinator will be the agent responsible for the price adjusting among the coalition members. Assigning co-ordinator's role to agents, is done in such a way that tries to evenly distribute the responsibilities by all the agents. Finally, it must be sent to all co-ordinators the information that they are co-ordinating a coalition, the composition of the coalition team, the individual prices proposed by each of the agents in that coalition and the best price achievable at this stage. In the limit, it will also be possible to have “coalitions” of one single agent.

- Market manipulation – the announcer sends to the co-ordinators the best price at this stage.
- Price adjusting – Each coalition co-ordinator's evaluates its possibility of improving the previous coalition's bid. In order to do that, he has to establish a negotiation with the coalition's partners as we will see later. If it is not possible to get a price lower than the best so far, the agent retires its coalition from the process by sending a specific message to the announcer. If a lower price is possible to be derived, the co-ordinator sends out the new bid to the announcer.
- Price selection – the announcer, after receiving all the answers from all the co-ordinators (either offering better conditions or quitting), evaluates the best offer and communicates the new best to all the active co-ordinators.

These last two steps are repeated until just one coalition remains active or some timeout is reached.

In face of the received bids, the announcer defines the possible coalitions and sends out to the co-ordinators the value of the best proposal. As an example, suppose that only two coalitions are approved: C1={A,B,C,D} and C2={B,C,D,E}. The chosen co-ordinators are respectively A and B. Therefore, they get the best value at this stage, say 80. An intra-coalition negotiation leads to a new value of 75 for coalition C1. This new value is communicated to the announcer that sends it again to the co-ordinators. A similar process leads coalition C2 to decrease its value to 70. In face of this new value, C1 gives up and C2 is chosen to perform the task with a cost of 70.The *intra-coalition* negotiation is also an interesting process. It will be done accordingly to the following algorithm:

- *1st step* – co-ordinator, after receiving the information about the coalition composition and the best offer so far, calculates the percentage of the initial cost that the team must decrease in order to reach a lower price bid.
- *2nd step* – co-ordinator sends to all the agents in the coalition (including itself) that figure (percentage).
- *3rd step* – agents respond to the previous message with either an acceptance or a rejection. In the latter case, the agents will inform about their minimal acceptable price.
- *4th step* – co-ordinator receives the answer from all the agents involved and calculates the new coalition bid. If some of the agents did not accept the proposed cost reduction and just send their minimum, but some other accepted it, a new percentage must be now calculated considering these minimum values. The new percentage for decreasing values must then be proposed again to the remaining agents (those that are ready to lower their bids) by going back to step 3. If there are no more accepting agents the algorithm steps forward to step 5.
- *5th step* – co-ordinator informs the announcer about the new coalition decision – a better offer or a withdraw.

Several interesting problems related with coalition formation have also been tackled by us [11] including situations where an Agent may belong to different coalitions. Here we have only included the flexible negotiation protocol for both helping on "intra" coalition bid generation as well as for comparative evaluation and appropriate selection of the best coalition bids.

3. Knowledge for Teamwork Coordination

One specific domain that has provided stimulating scenarios for experimenting co-operation among agents together with tight coordination, is robosoccer both in the hardware and simulation versions.

Robosoccer [21] [14], like other coordination demanding applications as it the case of rescue-like or battlefield combats, have been a permanent inspiration for developing both agent architectures and multi-agent coordination policies, suitable for efficient co-operation in achieving goals in adversarial environments.

The key aspect on developing a suitable agent architecture is always related with finding out a good trade-off between real-time, efficient, reactive agent behaviour and a rich, cognitive and flexible social deliberation capability.

This trade-off is important whenever we want an agent to perform both individually and as a responsible member of a team for achieving complex goals in dynamic environments.

Interesting mechanisms have been proposed [20] [21] for selecting pre-fixed multi-agent protocols available for all the elements in the team. Teamwork is there achieved through task decomposition and dynamic role assignment [21]. For those authors, team strategy takes advantage of formations using protocols for switching between them. Each formation assigns each agent a given role and protocols are suggested for enabling role exchange between the agents. In robotic soccer roles mainly correspond to specific positioning on the field.

Another implemented general model for teamwork is the STEAM (Shell for Teamwork) [23]. STEAM is based on the theory of joint intentions [7] but also on the SharedPlan theory [4]. STEAM uses joint intentions as the basis for teamwork but team members also build up a hierarchical structure of joint intentions, individual intentions and beliefs about the teammate's intentions [7].

The trade-off between reactivity (essential to cope with the real-time, dynamic, noisy environment) and cooperation (needed to enable team joint behaviour and to achieve overall team goals in the adversarial environment) is very difficult to achieve in the context of a general cooperative framework. We have proposed [13, 14] a new approach to agents' team coordination together with a method to balance reactivity and social behaviour. Our approach includes:

- Balancing social behaviour and reactivity through the use of a clear distinction between active and strategic situations;
- Situation based strategic positioning – a policy used to position the agents in situations classified as strategic situations;
- Dynamic role and positioning exchange – enabling agents to switch roles (agent's behaviors) and positioning;
- Formalization of what is a team strategy for a competition in this kind of domains, based on the concepts of tactics, formations and roles;

The proposed approach is based on the definition of a team strategy using the concepts of tactics, formations and roles. Agents' decision making is based on a clear distinction between strategic and active situations. Based on this distinction, agents use, for strategic behaviour, Situation Based Strategic Positioning and, for active behaviour, domain specific high-level and low-level skills. To improve the flexibility of the team, agents are also able to switch their positions and specific behaviors (roles), at run-time, on the field. This mechanism called Dynamic Positioning and Role Exchange (DPRE) is based on previous work by Peter Stone et al [20, 21] that suggested the use of flexible agent roles with protocols for switching among them. We have extended this concept and suggested that agents may exchange not only their roles (that correspond to agent types in our formulation) but also their positioning in the current formation if the utility of that exchange is positive..

Following our approach [13, 14], a team strategy is composed of a set of tactics and a set of possible agent types (defining agent behaviours at several levels). Each tactic is composed of several formations that are applied in different situations. Each formation is a flexible distribution of the agents on the field and implies the assignment of specific types to those agents. Each agent strategic position on the field is both dependent on the current situation and of the positioning assigned through the selected formation. This nested strategic and tactic knowledge representation enables the development of very flexible teams, capable of assuming social behaviour (in strategic situations) and a more reactive behaviour (in active situations). Moreover the distinction between strategic and active situations gives the agents more sophisticated means of achieving team global coordination. When the agent is not on a critical situation (one in which he will be engaged in active behaviour), it assumes that it is in a strategic situation and, therefore, it tries to position itself according to team formation.

Our Dynamic Positioning and Role Exchange mechanism is a policy enabling a better use of the team resources by exchanging agents positioning and role. Whenever it is better for the team global utility that two agents switch their respective locations, the exchange is performed.

4. Conclusions

We have here advocate that agents, placed in dynamic and sometimes adversarial environments, need specific characteristics enhancing their adaptive, negotiation and co-ordination capabilities.

Although not reported in the paper, we had the opportunity to benefit from many experiments done in a number of different practical applications - MACIV system for Agent-based distributed resources management, SMACE for agent-based Electronic Commerce, ForEV a platform for Virtual Enterprises Formation . Those experiments lead us to here propose multi-criteria qualitative negotiation, coalition formation with associated intra-coalition negotiation protocol and Q-learning based adaptive algorithms for agent-based systems. Moreover, again guided by our experiments, this time in the robosoccer domain, we briefly point out needed knowledge that has to be represented in order to facilitate flexible agents' teamwork.

Acknowledgements

Acknowledgements are due to A. P. Rocha, H. L. Cardoso, J. M. Fonseca and L. P. Reis, for their crucial contribution for the work here reported.

References

1. Cardoso H.L, Oliveira Eugénio, A Platform for Electronic Commerce with Adaptive Agents, in Agent-Mediated Electronic Commerce III, ed. F.Dignum and U.Cortés, pp.96-107, Lecture Notes in Artificial Intelligence 2003, Springer, 2001.
2. Faratin, P., C. Sierra and N.R. Jennings (1998), "Negotiation Decision Functions for Autonomous Agents", *International Journal of Robotics and Autonomous Systems*, 24 (3-4), pp. 159-182.
3. Georgeff M:P:, Lansky A:L:, Reactive reasoning and planing, in Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI'87), pp.677-682, Seattle, 1987.
4. Grosz B., and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86, 269—358, 1996
5. Guttman, R. H., Moukas, A. G., and Maes, P., "Agent-Mediated Electronic Commerce: A Survey", Knowledge Engineering Review, June 1998
6. Ingrand F., Rao A., Georgeff M., " An architecture for real-time reasoning system control", IEEE Expert 7(6), 1992
7. Levesque H. J., Cohen P. R. and Nunes J.. On acting together. *In Proceedings of the National Conference on Artificial Intelligence.* Menlo Park, California, AAAI Press, 1990.
8. R.Lewicki, D.Saunders, J.Minton, "Essentials of Negotiation", Irwin, 97.
9. Matos N., Sierra C., Jennings N., Determining successful negotiation strategies: An Evolutioanary approach, in Proceedings of the third International Conference on Multi-Agent Systems (ICMAS'98), Paris 1998.
10. Moore J., Implementation, Contracts and Negotiation in Environments with complete information, in Advances in Economic Theory, V.1, Cambridge University Press, 1992.
11. Oliveira Eugénio, Fonseca J.M., Garção A.S., MACIV: A DAI based resource management system, in Applied Artificial Intelligence, V.11,N.6. pp.525-550, Taylor&Francis, United Kingdom 1997.
12. Oliveira E., Rocha A.P., Agents advanced features for Negotiation in Electronic Commerce and Virtual Organisations Formation Process, in Agent Mediated Electronic Commerce: The European AgentLink Perspective, Ed. F.Dignum and C.Sierra, pp.78-97, LNAI 1991, Springer 2001.
13. Reis L.P., Lau N., Oliveira E., Situation based Strategic Positioning for Coordinating a Team of Homogeneous Agents, to appear in Lecture Notes of Computer Science, Springer 2001.
14. Reis L.P., and Lau N., FC Portugal Team Description: RoboCup 2000 Simulation League Champion, *In Peter Stone, Tucker Balch and Gerhard Kraetzschmar, editors, RoboCup-2000: Robot Soccer World Cup IV*, Springer Verlag, Berlin, 2001, to appear
15. Sandholm T., Distributed rational Decision Making, in Multiagent Systems: A Modern approach to Distributed Artificial Intelligence, ed. G.Weiss, pp.201-258, The MIT Press, 1999.
16. Sandholm T., Agents inElectronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation, in Autonomous Agents and Multi-Agent Systems, V.3, N.1, pp.73-96, March 2000, Kluwer A.P.
17. Sen S., Weiss G., Learning in Multiagent Systems, in Multiagent Systems: A Modern approach to Distributed Artificial Intelligence, ed. G.Weiss, pp.259-298, The MIT Press, 1999
18. Sierra C., Jennings N., Noriega P., Parsons S. "A framework for Argumentation Based Negotiation" (ATAL97) Intelligent Agents IV (M. P.Singh, A. Rao and M. J. Wooldridge, eds). Springer-Verlag LNAI 1365 (1998) pp 177-192.
19. Singh M., Rao A., Georgeff M., "Formal methods in DAI logic-based representation and reasoning", in Multiagent Systems: A Modern approach to Distributed Artificial Intelligence, ed. G.Weiss, pp.331-376, The MIT Press, 1999.

20. Stone P., *Layered Learning in Multi-Agent Systems*. PhD Thesis, School of Computer Science, Carnegie Mellon University, 1998
21. Stone P., Veloso M.,Task decomposition, Dynamic Role assignment and low bandwidth communication for real time strategic teamwork, AI110(2),pp.241-273, June,1999.
22. Sutton R.S. and Barto A.G., *Reinforcement Learning: An Introduction*, Cambridge: MIT Press, 1998.
23. Tambe M., Towards Flexible Teamwork, *Journal of Artificial Intelligence Research* 7, pp. 83-124, 1997
24. Vulkan, N. and Jennings, N. R., “Efficient Mechanisms for the Supply of Services in Multi-Agent Environments,” presented at 1st Int Conf. on Information and Computation Economies, Charleston, South Carolina, 1998.
25. Wellman,M. and Wurman, P. Market-aware Agents for a Multiagent World, in Robotics and Autonomous Systems Journal, 24: 115-125, 1998.
26. Wooldrige M., Intelligent Agents, in Multiagent Systems: A modern approach to Distributed Artificial Intelligence, ed. G.Weiss, pp.27-78, The MIT Press, 1999.
27. Wooldridge M., Jennings N. "Intelligent Agents: Theory and Practice", The Knowledge Engineering Review, 10(2), pp.115-152, 1995.

Towards Heterogeneous Agent Teams

Milind Tambe and David V. Pynadath

Computer Science Department and Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{tambe,pynadath}@isi.edu

Abstract. Agent integration architectures enable a heterogeneous, distributed set of agents to work together to address problems of greater complexity than those addressed by the individual agents themselves. Unfortunately, integrating software agents and humans to perform real-world tasks in a large-scale system remains difficult, especially due to two key challenges: ensuring robust execution in the face of a dynamic environment and providing abstract task specifications without all the low-level coordination details. To address these challenges, our Teamcore project provides the integration architecture with general-purpose teamwork coordination capabilities. We make each agent *team-ready* by providing it with a proxy capable of general teamwork reasoning. Thus, a key novelty and strength of our framework is that powerful teamwork capabilities are built into its foundations by providing the proxies themselves with a teamwork model called STEAM. While STEAM has earlier been demonstrated in domains involving homogeneous agent teams, its use in Teamcore proxies illustrates that teamwork models may also be applied in domains involving heterogeneous agents. Given STEAM, the Teamcore proxies addresses the first agent integration challenge, robust execution, by automatically generating the required coordination actions for the agents they represent. We can also exploit the proxies' reusable general teamwork knowledge to address the second agent integration challenge. Through *team-oriented programming*, a developer specifies a hierarchical organization and its goals and plans, abstracting away from coordination details. Our integration architecture enables teamwork among agents with no coordination capabilities, and it establishes and automates consistent teamwork among agents with some coordination capabilities. We illustrate how the Teamcore architecture successfully addressed the challenges of agent integration in two application domains: simulated rehearsal of a military evacuation mission and facilitation of human collaboration.

1 Introduction

An increasing number of agent-based systems now operate in complex dynamic environments, such as disaster rescue missions, monitoring/surveillance tasks, enterprise integration, and education/training environments. With this increasing population of available agents, we can expect another powerful trend: the reuse of specialized agents as standardized building blocks for large-scale systems [11, 13, 9]. This prediction is based on two observations. First, developers continue to construct software systems out of ever-larger reusable components, rather than as monoliths [23]. The reuse of agents as components is the next logical step — enabling a richer reuse mechanism than component-ware or application frameworks [13]. Second, there is the already growing trend of integration of agent components in cooperative information systems, networked embedded systems, and other software systems [10, 11, 22].

System designers can integrate these existing agents to construct new multi-agent systems capable of solving problems of greater complexity than those addressed by the individual agents themselves. Agent integration architectures enable a heterogeneous, distributed set of agents to work together to address such large-scale problems. Such integration architectures face an ever-increasing variety of available agents. In addition, as these agents move into more and more domains that require human interaction, these integration architectures must also tackle the coordination of people who exhibit a diversity and complexity beyond that of even software agents.

Unfortunately, integrating agents to perform real-world tasks in a large-scale system remains difficult. There are at least three key challenges that agent integration architectures face. First, it is difficult to ensure robust and flexible execution of the desired tasks. In addition to the risk of failures among individual agents, an integrated system also runs the risk of coordination breakdowns, due to (for instance) one agent's lacking key information known to the others. However, in an open environment, we cannot expect agents to come ready-made to avoid such breakdowns. Furthermore, even if an agent is capable of proper coordination (as is the case with people), it is often preferable for the architecture to take on some of this burden and free the agent or person to direct its resources to its individual tasks. Second, in general, building an integrated system to accomplish robust execution can require a potentially large number of coordination plans to cover all of the low-level coordination details. This problem is further exacerbated when the designer must create new plans for each new systemwide task or set of agents. Third, it is often difficult to locate and recruit relevant agents for integration in a distributed, open environment. There are other challenges as well (e.g., agent communication languages), but this article focuses on the three listed here.

We begin with a discussion of the coordination challenge in agent integration. To address this challenge, our architecture, called Teamcore[1], focuses on general-purpose teamwork capabilities. Existing theories of teamwork, such as joint intentions [3] and SharedPlans [7], provide an analytical framework for designing coordination behavior with strong guarantees. This research laid the theoretical groundwork for implemented systems that demonstrated the real-world utility of teamwork in designing robust organizations of agents that coordinate amongst themselves [25, 24, 12]. Based on these successful applications of teamwork to closed multiagent systems, the key hypothesis behind Teamcore is that teamwork among agents can enhance robust execution even among heterogeneous agents in an open environment. No matter how diverse the agents may be, if they act as team members, then we can expect them to act responsibly towards each other, to cover for each other's execution failures, and to exchange key information.

Therefore, our integration architecture is founded on powerful in-built teamwork capabilities. Essentially, the architecture enables teamwork among agents with no coordination capabilities, and it establishes and automates consistent teamwork among

[1] Teamcore derives its name from its encapsulation of "core team reasoning" as discussed later.

agents with some coordination capabilities, by providing them with a proxy capable of general teamwork reasoning. Each proxy contains a general teamwork model for such reasoning, which it uses to provide consistent *team readiness* to the heterogeneous agent it represents. Since team members behave responsibly towards each other, a team formed using such proxies can achieve its goals robustly, with agents automatically covering for failed teammates, supplying key information to help each other, etc. The novelty of our architecture stems from these in-built teamwork capabilities that provide the required robustness and flexibility in agent integration, without requiring modification of the agents themselves. This contrasts with other architectures such as OAA [18] that provide centralized facilitators, but require the hand-generated addition of such teamwork capabilities to the agents being integrated. The distributed nature of Teamcore also avoids any centralized bottlenecks and central points of failure.

We have implemented our Teamcore architecture using STEAM [24] as the proxies' teamwork model. STEAM provides a reusable, general-purpose teamwork module that encapsulates reasoning about common teamwork coordination, including contingencies in such coordination. STEAM has already proven effective in multiple coordination domains, so it forms a natural basis for providing normative teamwork capabilities in the more open, heterogeneous environments that Teamcore addresses. Given the STEAM module, the Teamcore proxies automatically generate the required coordination actions in executing their tasks. They communicate amongst themselves to ensure coherent execution of the tasks and to disseminate relevant information to the appropriate team members.

Because the Teamcore proxies automatically generate the required coordination actions in executing their plans, they shield the human developer from the second agent integration challenge of generating all of the low-level coordination details by hand. With the Teamcore architecture, we can exploit the proxies' reusable general teamwork knowledge to support abstract plan specification through *team-oriented programming*. Through team-oriented programming, a developer specifies a hierarchical organization and its goals and plans, abstracting away from coordination details. KARMA, our Knowledgeable Agent Resources Manager Assistant, can aid the developer in conquering the third agent integration challenge by locating agents that match the specified organization's requirements and assisting in allocating organizational roles to these agents.

Section 2 motivates the requirements for agent integration by describing the two multiagent domains to which we have applied the Teamcore architecture. Section 3 provides an overview of the Teamcore architecture. The next three sections describe team-oriented programming, STEAM, and Teamcore in more detail. Section **??** provides an initial evaluation of Teamcore. Section 8 compares other agent integration architectures related to Teamcore. Section 9 summarizes the contributions of the work presented here.

2 Motivation

This paper describes, illustrates, and discusses the Teamcore framework using two concrete examples — evacuation of civilians stranded in a hostile area and human collaboration — where we have successfully applied this framework. The rest of this section provides more detailed descriptions of these two domains.

2.1 Application 1: Evacuation Rehearsal

In the evacuation domain, the goal is an integrated system for simulated mission rehearsal of the evacuation of civilians from a threatened location. The system must enable a human commander to interactively provide locations of the stranded civilians, safe areas for evacuation, and other key points. A set of simulated helicopters should fly a coordinated mission to evacuate the civilians. The integrated system must plan routes to avoid known obstacles, dynamically obtain information about enemy threats, and change routes when needed. The following agents were available:

– **Quickset:** (from P. Cohen et al., Oregon Graduate Institute) Multimodal command input agents [C++, Windows NT] [4]
– **Route planner:** (from Sycara et al., Carnegie-Mellon University) Retsina path planner for aircraft [C++, Windows NT] [22]
– **Ariadne:** (from Minton et al., USC Information Sciences Institute) Database engine for dynamic threats [Lisp, Unix] [15]
– **Helicopter pilots:** (from Tambe, USC Information Sciences Institute) Pilot agents for simulated helicopters [Soar, Unix]

As this list illustrates, the agents are developed by different research groups, they are written in different languages, they run on different operating systems, they may be distributed geographically (e.g., on machines at different universities), and they have *no pre-existing teamwork capabilities*. There are actually 11 agents overall, including the Ariadne, route-planner, Quickset, and eight different helicopters (some for transport, some for escort). These agents provided a fixed specification of possible communication and task capabilities. Thus, the challenge in this domain lies in getting this diverse set of distributed agents to work together, without directly modifying the agents themselves.

2.2 Application 2: Assisting Human Collaboration

We have also applied Teamcore to assist human collaboration in our research team by automating many of our routine coordination tasks. Here, the agents to be integrated are members of our research group. The proxies know their users' scheduled meetings (by monitoring their calendars) and their whereabouts (e.g., whether they are working at their workstations). The Teamcore proxies must then assist in robust execution of team activities such as meetings. For example, if a user is still working at his/her workstation at the time of the meeting (e.g., to finish a paper), others should be automatically informed of an appropriate meeting delay. The overall system must also assign people to roles within team activities (e.g., selecting someone to give a presentation at a weekly research group meeting).

This system faces the daunting challenges of the users' heterogeneity (e.g., different presentation capabilities for different topics) and the larger scale of the team activities (e.g., each person is a member of multiple subgroups and has multiple meetings). In addition, the system cannot simply assign tasks for people, as it would the software agents of the evacuation domain. The system must also provide reliable communication with the users to perform these coordination tasks. One interaction mechanism available is the use of dialog boxes on the user's workstation display. Within our research group, five members currently have PDAs that the system can also exploit for interactions. In addition, the PDA can also provide location information if connected to a Global Positioning System (GPS) device (as in Figure 1). As a final means of communication, a proxy can send email to a project assistant or some other third party who can contact the user directly to pass on the message.

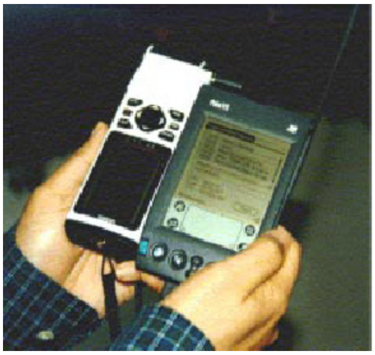


Fig. 1. PDA (Palm VII) with GPS device for wireless, handheld communication between proxy and user.

3 Overview of Teamcore

Figure 2 shows the overall Teamcore framework for building agent organizations. The numbered arrows show the typical stages of interactions in this system. In stage 1, human developers interact with a team-oriented programming interface (TOPI) to specify a team-oriented program, consisting of an organization and its team plans. TOPI communicates this specification to KARMA in stage 2. In stage 3, KARMA derives the requirements for roles in the organization, and searches for agents with relevant expertise (called *domain agents* in Figure 2). To this end, KARMA queries different middle agents, white pages (Agent Naming Service), etc. Once it has located these domain agents, KARMA further assists a developer in assigning agents to organizational roles.

Having thus fully defined a team-oriented program, the developer launches the Teamcore proxies that jointly execute the team plans of the team-oriented program. To perform the coordination necessary for this execution, the proxies broadcast information among themselves via multiple broadcast nets (stage 4). The Teamcore proxies execute the team plans and, in the process, also generate specific requests and process the replies of their domain agents (stage 5). KARMA also eavesdrops on the various broadcasts to monitor the Teamcore proxies' progress (stage 6), which it displays to the software developer for debugging purposes. All communication among Teamcore proxies, between a domain agent and its Teamcore proxy, and between a Teamcore proxy and KARMA currently occurs via the KQML agent communication language [5].

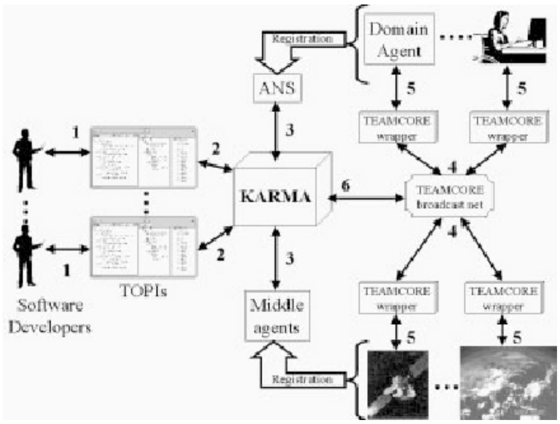


Fig. 2. Teamcore framework: Teamcore proxies for heterogeneous domain agents.

To ensure robust execution, the Teamcore architecture transforms agents of all types into a set of consistent team players. As described in Section 1 and as illustrated in Figure 2, we achieve this team readiness among heterogeneous agents by providing each agent with a Teamcore proxy. The distributed Teamcore proxies, based on the Soar [19] rule-based integrated agent architecture, execute their joint plans in a distributed fashion and coordinate as a team during this execution.

In the following, Section 4 describes team-oriented programming. Section 5 describes the contents of the STEAM module that encodes the definition of team readiness that the Teamcore proxies use in coordination. Section 6 describes how the Teamcore proxies apply this definition in coordinating the actions of their domain agents.

4 Constructing Team Plans and Organization

The developer begins specifying an organization of interest via team-oriented programming, in which the developer specifies three key aspects of a team: a team organization hierarchy, a hierarchy of reactive team plans, and assignments of agents to plans. The team organization hierarchy consists of roles for individuals and for groups of agents. For example, Figure 3-a illustrates a portion of the organization hierarchy of the roles involved with the evacuation scenario (described in more detail in Section 2.1). Each leaf node corresponds to a role for an individual agent, while the internal nodes correspond to (sub)teams of these roles. *Task Force* is thus the highest level team in this organization, while *Orders-Obtainer* is an individual role.

The second aspect of team-oriented programming is specifying a hierarchy of reactive team plans. While these reactive team plans are much like reactive plans for indi-

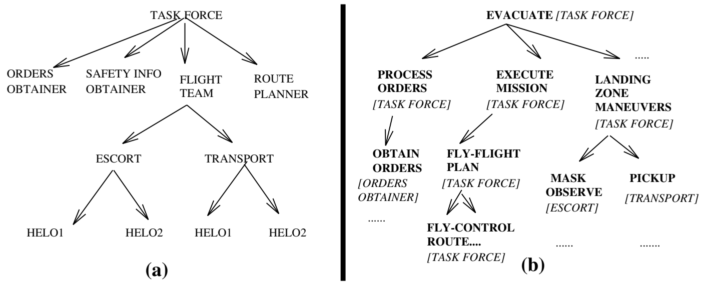

TASK FORCE
ORDERS OBTAINER
SAFETY INFO OBTAINER
FLIGHT TEAM
ROUTE PLANNER
ESCORT
TRANSPORT
HELO1
HELO2
HELO1
HELO2
(a)
EVACUATE [TASK FORCE]
PROCESS ORDERS [TASK FORCE]
EXECUTE MISSION [TASK FORCE]
LANDING ZONE MANEUVERS [TASK FORCE]
OBTAIN ORDERS [ORDERS OBTAINER]
FLY-FLIGHT PLAN [TASK FORCE]
FLY-CONTROL ROUTE.... [TASK FORCE]
MASK OBSERVE [ESCORT]
PICKUP [TRANSPORT]
(b)

Fig. 3. The evacuation scenario: (a) Partial organization hierarchy; (b) Partial team plan hierarchy.

vidual agents, the key difference is that the team plans explicitly express joint activities. The reactive team plans require that the developer specify the: (i) initiation conditions under which the plan is to be proposed; (ii) termination conditions under which the plan is to be ended, specifically, the conditions when the reactive team plan is achieved, irrelevant or unachievable; and (iii) team-level actions to execute as part of the plan. Figure 3-b shows an example from the evacuation scenario (please ignore the bracketed names for now). Here, high-level reactive team plans, such as **Evacuate**, typically decompose into other team plans, such as **Process-orders** (to interpret orders provided by a human commander). **Process-orders** is itself achieved via other sub-plans such as **Obtain-orders**. The precise detail of how to execute a leaf-level plan such as **Obtain-orders** is left unspecified — thus both simplifying the specification, and allowing for the use of different agents to execute this plan.

The software developer must also specify domain-specific plan-sequencing constraints on the execution of team plans. In the example of Figure 3, the plan **Landing-Zone-Maneuvers** has two subplans: **Mask-Observe** which involves observing the landing zone while hidden, and **Pickup** to pick people up from the landing zone. The developer must specify the domain-specific sequencing constraint that a subteam assigned to perform **Pickup** cannot do so until the other subteam assigned **Mask-Observe** has reached its observing locations.

The third aspect of team-oriented programming is the assignment of agents to plans. This is done by first assigning the roles in the organization hierarchy to plans and then assigning agents to roles. Assigning only abstract roles rather than actual agents to plans provides a useful level of abstraction: new agents can be more quickly (re)assigned when needed. Figure 3-b shows the assignment of roles to the reactive plan hierarchy for the evacuation domain (in brackets adjacent to the plans). For instance, *Task Force* team is assigned to jointly perform **Evacuate**, while the individual *Orders-obtainer* role is assigned to the leaf-level **Obtain-orders** plan. Associated with such leaf-level plans are specifications of the requirements to perform the plan. For instance, for **Obtain-orders**, the requirement is to interact with a human. A role inherits the requirements from each plan that it is assigned to. Thus, the requirements of a role are the union of the requirements of all of its assigned plans. Once KARMA derives requirements for individual roles in the organization, it searches for agents whose capabilities match those requirements, and helps the software developer in assigning agents to roles in the organization. This completes the team-oriented program, and STEAM-based Teamcore proxies enable smooth execution of this program.

The real key here is what is *not* specified in the team-oriented program: details of how to realize the coordination specified, e.g., how members of *Task Force* should jointly execute **Evacuate**. Thus, for instance, the developer does not have to program any synchronization actions, because STEAM generates them automatically, as described in Section 5. Thus, during execution, synchronization actions among members of *Task Force* are automatically enforced, both with respect to the time of plan execution and the identity of the plan (i.e., all members will choose the same plan out of a set of multiple candidates). Similarly, there is no need to specify the coordination actions for coherently terminating reactive team plans; such actions are automatically executed by the proxies due to STEAM. Domain-specific plan-sequencing constraints, such as the one between **Mask-Observe** and **Pickup** discussed above, are also automatically enforced.

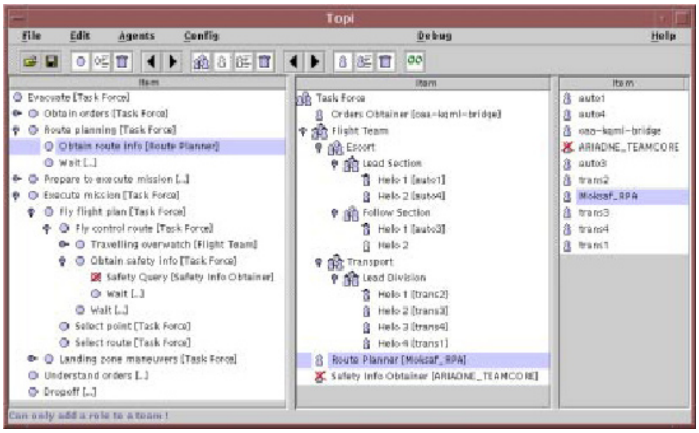
Likewise, the developer does not have to specify how team members should cover for each other in case of failures; rather, the proxies use STEAM for monitoring and repair to automatically replace fallen teammates. The team-oriented programming phase automatically generates the required capabilities for each role in the organization, as well as the capabilities of each available agent. If an agent should fail during execution, the proxies can follow STEAM to automatically find any available replacements for each of its roles based on these capability requirements.

Figure 4 shows a sample screenshot from TOPI used in programming the evacuation scenario, where the three panes correspond to the plan hierarchy (left pane), organization hierarchy (middle pane), and the domain agents (right pane). The left pane essentially reflects the diagram 3-b, e.g., *Task Force* has been assigned to execute **Evacuate**. Associated with each entity are its properties, e.g., associated with each plan are its coordination constraints, preconditions, assigned subteam, and so on.

5 STEAM: Making Heterogeneous Agents Team Ready

Each Teamcore proxy contains the STEAM domain-independent teamwork module, responsible for Teamcore's teamwork reasoning [25, 24]. Appendix A provides a detailed specification of the STEAM teamwork model, that enables it to generate teamwork actions in a domain-independent fashion. At its core, STEAM is based on the *joint intentions* theory [17, 3]; but it also parallels and in some cases borrows from the *SharedPlans* theory [6, 7, 8]. Thus, while STEAM uses joint intentions as the basic building block of teamwork, as in the SharedPlan theory, team members build up a complex hierarchical structure of joint intentions, individual intentions and beliefs about others' intentions. In STEAM, communication is driven by commitments embodied in the joint intentions theory — team members may communicate to attain mutual belief while building and disbanding joint intentions. Thus, joint intentions provide STEAM a principled framework for reasoning about communication, providing significant flexibility.

STEAM's key novelty is that it is one of the first teamwork models to be applied and reused in real-world domains. Such illustration of reuse provides an empirical demon-



```
Evacuate [Task Force]
  Obtain orders [Task Force]
  Route planning [Task Force]
    Obtain route info [Route Planner]
    Wait [...]
  Prepare to execute mission [...]
  Execute mission [Task Force]
    Fly flight plan [Task Force]
      Fly control route [Task Force]
        Travelling overwatch [Flight Team]
        Obtain safety info [Task Force]
          Safety Query [Safety Info Obtainer]
          Wait [...]
        Wait [...]
      Select point [Task Force]
      Select route [Task Force]
    Landing zone maneuvers [Task Force]
  Understand orders [...]
  Dropoff [...]
```

```
Task Force
  Orders Obtainer [oaa-kqml-bridge]
  Flight Team
    Escort
      Lead Section
        Helo 1 [auto1]
        Helo 2 [auto4]
      Follow Section
        Helo 1 [auto3]
        Helo 2
    Transport
      Lead Division
        Helo 1 [trans2]
        Helo 2 [trans3]
        Helo 3 [trans4]
        Helo 4 [trans1]
  Route Planner [Moksaf_RPA]
  Safety Info Obtainer [ARIADNE_TEAMCORE]
```

```
auto1
auto4
oaa-kqml-bridge
ARIADNE_TEAMCORE
auto3
trans2
Moksaf_RPA
trans3
trans4
trans1
```

```
Can only add a role to a team!
```

Fig. 4. TOPI snapshot from generating team-oriented program for the evacuation scenario.

stration of cross-domain general teamwork principles. STEAM's operationalization of teamwork theories in complex, real-world domains has illustrated the utility of these theories in building practical systems. In addition, it has also shown the shortcomings of these theories, which STEAM addresses. One key illustration is in STEAM's detailed attention to communication overheads and risks, which can be significant. STEAM integrates decision theoretic communication selectivity — agents deliberate upon communication necessities vis-a-vis incoherency in teamwork. This decision theoretic framework thus enables improved flexibility in communication in response to unexpected changes in environmental conditions. STEAM also aids the team in recovering from individual team member's failure to perform tasks: it provides detailed specification for such recovery that have not been provided in previous work. To this end, STEAM facilitates monitoring of team performance by exploiting explicit representation of team goals and plans. If individuals responsible for particular subtasks fail in fulfilling their responsibilities, or if new tasks are discovered without an appropriate assignment of team members to fulfill them, team reorganization can occur. Such reorganization, as well as recovery from failures in general, is also driven by the team's joint intentions.

The STEAM specification is operationalized as a set of rules (some example rules are shown in appendix B). The STEAM specification (and its rules) can be divided into three categories:

Coherence preserving rules require team members to communicate with each other to ensure coherent initiation and termination of team plans. Coherent initiation ensures that all members of the team begin joint execution of the same team plan at the same time. Therefore, these rules prevent a helicopter from flying to its destination before all the other members of its flight team are ready to begin as well. Coherent termination requires that a team member inform others if it uncovers crucial information. We define "crucial information" as any condition that indicates that the team plan is achieved, unachievable, or irrelevant. For instance, the rules prescribe that anyone who is going to be late for a meeting must notify the other attendees, since the achievability of the meeting is now threatened.

Monitor and repair rules ensure that team members make an effort to observe the performance of their teammates, in case any of them should fail. If a critical team member (or subteam) should fail, we ensure that a capable team member (or subteam) takes over the role of the failed agent. For instance, the presenter at a research group meeting has a critical role with respect to the corresponding team plan, since without a presenter, the meeting will fail. Therefore, these rules specify that the team continuously monitor the presenter's ability to fulfill this role. If the presenter is unable to attend, these rules require that the team find some other capable team member to step in and give the presentation instead.

Selectivity-in-communication rules use decision theory to weigh communication costs and benefits to avoid excessive communication in the team. We thus ensure that the team performs coordination actions whose value in achieving coherent behavior outweighs the cost of communication. For instance, in the evacuation domain, communication is moderately expensive, due to the risk of enemy eavesdropping. Therefore, these rules would prescribe communication only when there is a sufficiently high likelihood and cost of miscoordination (e.g., transport helicopters arriving at rendezvous point at a different time from their escorts). Communication is much less costly in the human collaboration domain; however, the likelihood of miscoordination is also much lower, since the human team members perform some coordination actions themselves. For instance, the rules would *not* require communication to initiate a meeting plan, since all of the attendees have already entered the meeting into their calendar programs. On the other hand, the rules do require communication if an attendee is unable to arrive on time, since the other attendees are unlikely to know this information without any communication.

STEAM's 300 Soar rules are available in the public domain and have proven successful in several different domains reported in the literature. The novelty in the current work lies in the extensions to STEAM that enable the application of its rules to a much broader class of agents and problem domains.

6 Teamcore's Interface with Domain Agents

In previous work [25, 24], STEAM resided directly in the domain agent's knowledge base, which is often difficult (if not impossible) to implement in an open, heterogeneous environment. By placing STEAM's teamwork knowledge (rules) in a separate Teamcore proxy, we no longer need to modify code in the domain agent. However, the Teamcore proxy must now contain an interface module for communication with the domain agent, as illustrated in Figure 5. In particular, the STEAM rules enable the Teamcore proxies to automatically communicate with each other to maintain team coherence and recover from member failures. In contrast, the interface module enables a Teamcore proxy to communicate with its domain agent, by translating the state of the team's execution into individual tasks and monitoring requests.

The Teamcore proxy generates task requests according to the role assigned to its corresponding domain agent in the organization. If the overall state of the team's exe-

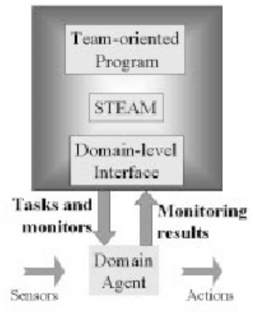


Fig. 5. Reasoning components of a Teamcore proxy and interactions with domain agent.

cution requires that a domain agent now perform a particular task, its Teamcore proxy generates the appropriate request message, based on the domain agent's interface specification and the proxy's knowledge of the state of the team plan. The proxy's adherence to the STEAM rules ensures that its beliefs about the state of the team plan agree with those of its teammates. Thus, the proxy is sure that its domain agent will perform the requested task in synchronization with its teammates. The domain agent can then process the resulting task request, without necessarily being burdened with understanding the larger team context.

The domain agent returns any result it may produce to its Teamcore proxy. The proxy may then communicate the result to its teammates, as mandated by STEAM's coherence-preserving rules. Again, the domain agent need not know anything about the overall team context. In the case of a simple agent that provides responses to a fixed set of queries, it sees only a request from its Teamcore proxy that it processes and responds to, just as it would for any other individual client. However, the result of the domain agent's actions still produce the desired teamwide effects, since the Teamcore proxy forwards the result to those teammates to whom the new information is relevant.

The Teamcore proxies generate monitoring requests in a similar fashion, except that multiple team members may perform the same monitoring task without regard to any assigned roles within the organization. Thus, multiple proxies may send requests to their corresponding domain agents for more robust monitoring. One key interesting issue in this architecture is that it is often the domain agent, and not the Teamcore proxy, that has access to information relevant to the achievement, irrelevance, and unachievability of the team plans (e.g., an information-gathering agent can search a database for known threats to a team of helicopters). Yet, only the Teamcore proxy knows the current team plans, so the domain agent may not know what observations are relevant (e.g., threats to the helicopters are relevant), necessitating communication about monitoring. For each team plan, the Teamcore proxies already maintain the termination conditions — conditions that make the team plan achieved, irrelevant, or unachievable. Each Teamcore proxy also maintains a specification of what its domain agent can observe. Thus, if a domain agent can observe conditions that reflect the achievement, irrelevance, or unachievability of a team plan, then the Teamcore proxy automatically requests it to monitor any change in those conditions. The response from the domain agent may be communicated with other Teamcore proxies, through the usual STEAM procedures.

The Teamcore proxies can similarly translate STEAM's monitor and repair rules into appropriate messages for the domain agents. For instance, in the human collaboration domain, each proxy monitors its user's ability to attend the meeting on time, perhaps asking the user directly. If the user responds that s/he is unable to attend, the proxy follows the STEAM rules and automatically forwards this information to the rest of the team. If the user fills a critical role in the meeting plan (e.g., s/he is the presenter), then the team must repair the plan before proceeding. The proxies, again following the STEAM rules, first determine whether their users have the capability of taking on the role, perhaps by asking directly. Finally, the proxies follow the STEAM repair rules to fill the role with one of the users whom they determine to be capable and then notify the selected user.

7 Preliminary Evaluation

Our application of the Teamcore framework to the two application domains described in Section 2 has provided rich testbeds for evaluating the architecture's ability to successfully coordinate agents. Section 7.1 discusses the Teamcore architecture's ability to support robust coordination within the two domains. Section 7.2 discusses the advantages that team-oriented programming provided in integrating the agents in the two domains.

7.1 Evaluation of Robust Execution of Team Plans

One key aspect of evaluation is robustness, one of the motivations for the teamwork foundations of Teamcore. In both application domains, we used our Teamcore framework to specify the necessary team-oriented program, assign domain agents, and launch their proxies, which then successfully and robustly executed the team-oriented program. The overall system runs successfully, continuing even in the face of software failures in individual agents; instead, the Teamcore proxies of the remaining agents try to substitute another agent with relevant expertise if necessary (the maintenance-and-repair rules of STEAM) and/or show graceful degradation. For instance, if the route planner or its Teamcore proxy were to suddenly crash, the system does not halt. Instead, it reverts back to using straight-line paths.

In the evacuation domain, we built an agent organization from the 11 available domain agents (listed in Section 2.1, with eight separate helicopter pilot agents). The team-oriented program contained 43 reactive team plans. KARMA located the domain agents based on the team-oriented program and the specified organization hierarchy (although, this particular research collaboration was pre-arranged with the other groups). The Teamcore proxies then successfully executed the team-oriented program. In other words, they communicated with each other when appropriate and generated the correct tasking and monitoring requests for the domain agents. To demonstrate the robustness of the resulting agent organizations, the Teamcore-based system for evacuation rehearsal has often been demonstrated live outside our laboratory.

In the human collaboration domain, as of the writing of this paper, the proxies have run 24 hours/day, 7 days/week for three months.[2] We have designed, implemented, and deployed proxies to coordinate the activities for fifteen members of our research division. Here, the team-oriented program contains 15 reactive team plans, but each team member has multiple instantiations of many of these plans. For instance, there are several different **Successful-meeting** team plans active in parallel, one for each meeting the user has scheduled in the coming week. Figure 6 plots the number of daily messages exchanged by the proxies while coordinating these plans. The size of the counts demonstrates the large amount of coordination actions necessary in managing all of the plans, while the high variability of the daily count illustrates the dynamic nature of the domain.

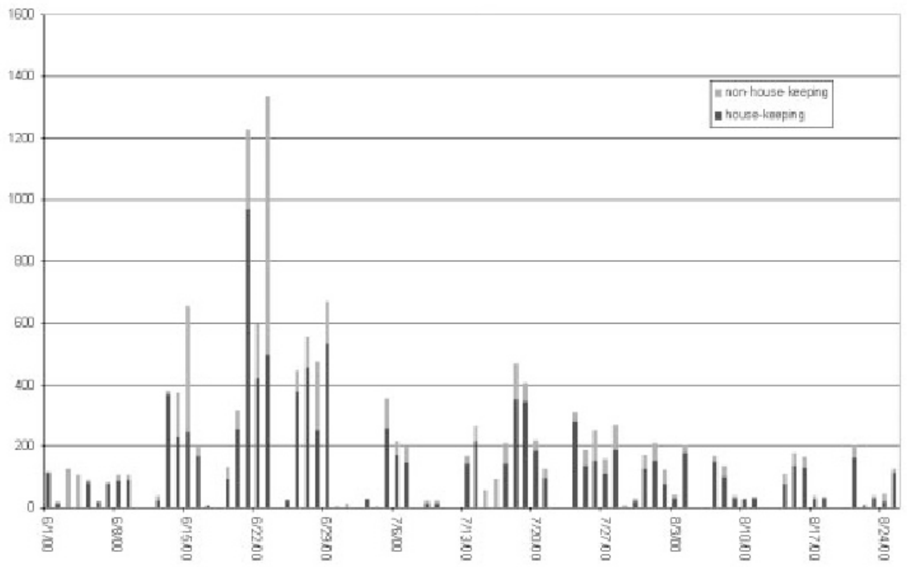


Fig. 6. Number of daily coordination messages exchanged by proxies over three-month period.

The distributed Teamcore architecture is well-suited to this domain, since people can maintain control of their own proxies, rather than centralizing the control and meeting information. Each proxy serves its user's interests in dealing with team-level activities. Thus, the proxy reasons about the user's willingness to accept joint activities and assigned roles within those activities. This reasoning requires that the proxies monitor

[2] These experimental runs are occasionally interrupted for bug fixes and enhancements.

their users' state and make decisions (possibly without any user input) about what they should report about that state to the team in the service of team goals.

The proxy often needs to interact with its user, whether to inform the user of a change in the status of a joint activity (e.g., a rescheduled meeting) or to ask for information before acting on a joint activity (e.g., whether the user wants a meeting delayed). As described in Section 2.2, the proxy exploits the user's workstation displays, any available PDAs, and email to third parties. The workstation display provides a simple Graphical User Interface (GUI) that allows the user to view what joint activities the proxy is currently monitoring. The GUI allows the user to initiate certain actions without waiting for the proxy to make them autonomously. The proxy displays the user's schedule and any informative messages on the PDA. The proxy can also pop up a dialog box on the PDA for feedback.

7.2 Evaluation of Team-Oriented Programming

One key dimension of the benefits of Teamcore proxy's in-built teamwork capabilities is in the abstraction provided by team-oriented programming. One key alternative to such an in-built teamwork model is reproducing all of Teamcore's capabilities via domain-specific coordination plans. In such a domain-specific implementation, about 10 separate domain-specific coordination plans would be required for each of the 40 team plans in Teamcore [24]. That is, we would require potentially hundreds of domain-specific coordination plans to reproduce Teamcore's capabilities to coordinate among each other for this domain alone. In contrast, with Teamcore, no coordination plans were written for inter-Teamcore communication. Instead, such communications occurred automatically from the specifications of team plans. Thus, it would appear that Teamcore has significantly alleviated the coding effort for coordination plans.

The Teamcore proxies' use of STEAM's selectivity in communication provides the additional benefit of automatically minimizing the amount of communication needed for proper coordination. Figure 7 shows the number of messages exchanged over time in different runs. The X axis measures the time elapsed while the Y axis shows the cumulative number of messages exchanged on a *log scale*. The "normal" run shows the number of messages typically exchanged among the Teamcore proxies for the evacuation scenario with time. The key here is that these approximately 100 messages are automatically generated by the Teamcore proxies. The "cautious" run shows the number of messages (approximately 1000) exchanged among the Teamcore proxies without the decision-theoretic communication selectivity in Teamcore, illustrating both the overhead reduction via such reasoning and the difficulty of hand-coding coordination (simple hand-coded coordination may lead to significant overheads). Finally, the "failure" run shows the messages exchanged among the proxies if the Ariadne agent were to crash unexpectedly (in the course of a "normal" run). To compensate for such failure, there is an initial increase in the total messages; but once the proxies compensate for the failure, fewer messages are exchanged, so that the total messages in the "failure" and "normal" runs is roughly the same.

Team-oriented programming also simplifies organizational modifications. Our framework appears to facilitate changes to the team, at least compared with the alternative of domain-specific coordination. For instance, the route planner was the last addition to the

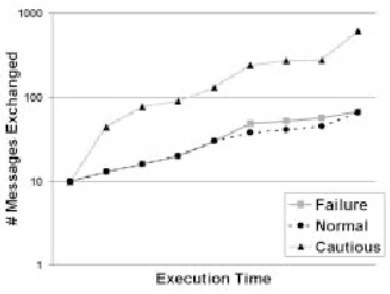


Fig. 7. Comparison of messages exchanged.

team. It required few modifications to the team-oriented program. To extend the organizational hierarchy, we simply added the route planner as a member of *Task Force*. We then added the **Process-Routes** branch of the plan hierarchy to allow for route planning. This branch involves very simple plans where the Teamcore proxy submits a request for planning a particular route, waits for the reply by the route planner, and then communicates the new route to the other team members according to STEAM's coherence-preserving rules. Again, no new coordination rules were required. It is similarly easy to modify the organization in the human collaboration example, where members join and leave, and teams form and dissolve.

8 Related Work

Section 1 briefly discusses Teamcore's relationship with centralized integration architectures such as OAA [18]. The Adaptive Agent Architecture (AAA) [16] provides a more distributed extension to OAA, allowing for teamwork among the facilitators. However, AAA does not provide teamwork among the agents themselves, thus limiting the robustness it can guarantee for the integrated system. Another related system, the RETSINA multi-agent infrastructure framework [22], is based on three different types of interacting agents: (i) interface agents; (ii) task agents; and (iii) information agents. Middle agents allow these various agents to locate each other. This effort appears quite complementary to Teamcore. Indeed, KARMA can use RETSINA middle agents for locating relevant agents, while infrastructural teamwork in Teamcore may enable the different RETSINA agents to work in teams.

COLLAGEN [20] models dialogue between a user and an agent — a form of joint activity — based on the SharedPlans [7] model of joint action. COLLAGEN has been previously compared to STEAM, and, like Teamcore, it provides a fairly clean separation between the teamwork layer and the problem-solving layer of the agent. However, COLLAGEN targets wrapping only a single agent for collaboration with a user, so that the issue of constructing and programming a team of agents and humans is not relevant.

Other than COLLAGEN, few other agent integration frameworks explicitly address the possibility of integrating people within the multiagent system. The rest of this section describes some integration frameworks that have been applied only to software agents, but that are still relevant to our Teamcore architecture. Tidhar [26, 27] used the term "team-oriented programming" to describe a conceptual framework for specifying team behaviors based on mutual beliefs and joint plans, coupled with organizational structures. This framework forms the basis of an implementation based on the dMars agent architecture [28]. In Tidhar's framework, the organizational hierarchy ensures that only appropriate agents (e.g., team leaders) fill specific roles offering certain authority or privilege. Tidhar describes how one can automatically unfold team plans into plans for individual agents containing communicative acts that ensure rudimentary coordination. His framework also addressed the issue of team selection [29] — team selection matches the "skills" required for executing a team plan against agents that have those skills. While many of the features of Tidhar et al.'s conceptual and implemented frameworks are important in the context of Teamcore, the critical issue of agent reuse, particularly involving heterogeneous (non-dMars) agents, is not given much attention. Thus, proxies with monitoring, tasking, and plan alteration capabilities and an agent resources manager such as KARMA for locating agents are novel in our framework. Furthermore, Teamcore's flexibility of reorganization and communication selectivity (through STEAM) does not seem to be part of the abstract team layer of Tidhar's framework.

Jennings's GRATE* [12] work also uses a teamwork module, one that has been previously compared to STEAM. GRATE* implements a model of cooperation based on the joint intentions framework, similarly used by STEAM. Each agent has its own *cooperation level* module that negotiates involvement in a joint task and maintains information about its own and other agents' involvement in joint goals. Regarding the specific issue of agent reuse, GRATE* separates the teamwork layer from the individual problem-solving layer of an agent. However, Teamcore's STEAM module allows teamwork to a deeper level than the single joint goal and plan in GRATE*. The more complex nature of the teams and team tasks in Teamcore has led us to explicitly focus on team-oriented programming and to explore several novel issues (e.g., automatic generation of monitoring conditions) that GRATE* does not address. STEAM also provides capabilities for role substitution in repairing team activity, a capability not available in GRATE*. Furthermore, GRATE* also does not address the issues of building team-oriented programs to specify agent organizations and KARMA-like agent resources manager to aid in building and monitoring such programs.

The ADEPT architecture for modeling business processes [14] allows a more flexible, hierarchical team organization than GRATE*. ADEPT consists of multiple *agencies*, each containing a *responsible agent*, which handles communication and interaction with other agencies. Various responsible agents maintain each agency's "capabilities", avoiding the use of a central facilitator or broker. A task is "contracted out" to an agency that has the capabilities to perform that task. As with GRATE*, ADEPT provides a fairly clean interface between the individual task-achieving agents and the social level. However, the ADEPT framework does not seem to address the issue of agent reuse directly, although the architecture itself could potentially incorporate heterogeneous agents. Also, ADEPT does not provide an explicit model of teamwork, such as that based on joint plans/intentions; instead, the basis of collaboration seems more closely related to the notion of *social commitment* [2].

Singh has proposed an abstract framework for coordinating heterogeneous agents [21]. Singh's model represents planned activity via finite-state automata (abstracting away the internal workings of the agents), where transitions represent external actions or events. The coordination service maintains knowledge of individual agents' actions as well as the overall joint plan and, upon receiving a request to perform an action, informs the appropriate agents as to whether an intended action should be executed, delayed, or omitted so as to fit with the joint activity of other agents. Singh's model does not address many of the issues of teamwork; however, it provides a potentially useful tool which could augment the joint plan framework of Teamcore with a language for specifying flexible, coordinated interactions at an abstract level.

Like the STEAM rule module within Teamcore, the COOL coordination framework [1] also focuses on general-purpose coordination by relying on obligations among agents. However, it explicitly rejects the notion of joint goals and joint commitments. It would appear that individual commitments in COOL would be inadequate in addressing some teamwork phenomena, but further work is necessary in understanding the relationship between COOL and Teamcore.

9 Summary

The two application domains tackled in this work, as well as most other real-world domains, present unique agent integration challenges of heterogeneity of agents (both software and human), number of the joint activities, complexity of the properly coordinated behavior, etc. While no previous agent integration architecture has yet resolved all of these challenges simultaneously, the Teamcore architecture takes a significant step forward. Teamcore's success in its two widely disparate application domains demonstrates the power and generality of the overall framework and provides strong evidence for its underlying hypotheses.

The primary hypothesis behind the Teamcore architecture and a key lesson learned from its success is that an agent integration infrastructure based on sound principles of agent coordination can automate robust coordination among distributed, heterogeneous agents. The Teamcore proxies' teamwork model proved sufficient in enabling robust coordination among the agents in both domains. The proxies' ability to reuse the same general-purpose rules to accomplish this robustness, despite the vast differences between the two domains, demonstrates the effectiveness of this teamwork knowledge.

In addition, by using the separate Teamcore proxies to perform the coordination, we are able to incorporate the domain agents without modifying them. This is especially important in an open environment, where our access to the internals of the agents is often minimal. Thus, the proxies' teamwork knowledge succeeded in coordinating agents that were essentially black boxes and that spanned the entire range of coordination capabilities, from the team-ready humans to the software agents completely incapable of coordination.

Teamwork also provides a useful layer of abstraction for coordinating heterogeneous agents. Once the agents become good team players (through their proxies), we are free to design at the level of team-oriented programming. We have constructed organizations using team members that covered the spectrum of agenthood, from databases capable of answering KQML queries, to people capable of multiple, parallel tasks and modes of interaction. Fortunately, in designing these organizations, one can ignore the details about the agents themselves. The developer instead thinks in terms of relevant joint activities: which agents are working together, what is the task they are performing, and who plays what roles. The proxies make the abstract specification of these joint activities operational by using their teamwork model to fill in the details about what and when messages would actually go among the agents.

Teamwork provides a sound basis for a coordination architecture, but it is important that the teamwork be flexible, to maximize the architecture's applicability across multiple domains. For instance, in the evacuation domain, where the domain agents had no coordination knowledge themselves, the Teamcore proxies were completely responsible for synchronizing the actions of the agents. The proxies would then be sure to communicate before initiating critical plans when miscoordinated execution among the various domain agents was sufficiently likely and costly. On the other hand, in the human collaboration domain, the domain agents are people who are already very capable of coordinating themselves. Thus, there was less of a burden on the Teamcore proxies to synchronize certain team plans, which then had a low cost of miscoordinated execution in the team-oriented program. In addition to flexibility in coordination specification, the coordination architecture must provide flexibility in execution. The teamcore architecture demonstrated the value of its flexibility in successfully supporting the two vastly different domains described in this paper.

In conclusion, the Teamcore architecture provides a novel means for integrating distributed, heterogeneous software agents and humans. Its use of a general-purpose teamwork model supports abstract specification of coordination behavior, robust execution of that behavior, and adaptation to world dynamics and heterogeneity. In addition, the proxy-based architecture supports coordination in an open environment where agent modifications may not always be possible. The success of the Teamcore architecture in its two application domains encourages us to continue expanding its capabilities and applications. We will continue running the proxies for the human collaboration domain, but we also plan to expand the number and types of team activities that they manage. We also plan to integrate more and more of our fellow researchers, as well as additional software agents, into the agent organization. As the size and complexity of the organization grows, the architecture will have to address new issues deriving from this scale-up (e.g., conflicts between team activities). We believe that the architecture's success will extend to many more real-world domains as well, providing a powerful framework for agent integration beyond that currently possible with existing technology.

Acknowledgements

This research was supported by DARPA Award no. F30602-98-2-0108 and managed by the AFRL/Rome research site. We thank Hans Chalupsky, Phil Cohen, Yolanda Gil, Craig Knoblock, Steve Minton, and Katia Sycara for contributing agents used in the work described here.

Appendix A: Detailed STEAM Specification

The pseudo-code described below follows the description of STEAM provided in this article. It is based on execution of hierarchical team reactive plans (as specified in a team-oriented program). All reactive-plans in the hierarchy execute in parallel, and hence the "in parallel" construct. The comments in the pseudo code are enclosed in /* */. The terminology is first described below, to clarify the pseudo-code.

- *Execute-Team-Reactive-Plan*(α, Θ, **C**, {ρ1, ρ2,...,ρn}) denotes the execution of a team reactive-plan α, by a team Θ, given the context of the current intention hierarchy **C**, and with parameters ρ1, ρ2...ρn.
- $[\alpha]_\Theta$ denotes the team Θ's joint intention to execute α.
- **EU** denotes computation to determine expected utility of an action, e.g., communication action.
- **status**($[\alpha]_\Theta$, *STATUS-OF-*α) denotes the status of the joint intention $[\alpha]_\Theta$, whether it is mutually believed to be achieved, unachievable or irrelevant.
- **satisfies** (Achievement-conditions(α),*f*) denotes that the fact *f* satisfies the achievement conditions of the team reactive-plan α; similarly with respect to unachievability and irrelevancy conditions.
- *Communicate*(terminate-jpg(α),*f*,Θ) denotes communication to the team Θ to terminate Θ's joint commitment to α, due to the fact f.
- Update-state (**team-state**(Θ), *f*) denotes the updating of the team state of Θ with the fact f.
- Update-status($[\alpha]_\Theta$) denotes the updating of the team reactive-plan α with its current status of achievement, unachievability or irrelevancy.
- **Agent**(α) is the individual agent or team executing reactive-plan α.
- **actions**(α) denote the actions of the reactive-plan α.
- **teamtype**(ψ) is a test of whether the agent ψ is a team or just one individual.
- **self**(ψ) is a test of whether the agent ψ denotes self.
- **agent-status-change**(μ) denotes change in the role performance capability of agent or subteam μ.
- *Execute-individual-Reactive-Plan*(ψ, self, **C**, {ρ1, ρ2,...,ρn}) denotes the execution of an individual reactive-plan ψ by self, given the context of the current intention hierarchy **C**, and with parameters ρ1, ρ2...ρn.

Team Reactive-Plan Execution

```
Execute-Team-Reactive-Plan($\alpha$, $\Theta$, C, {$\rho$1, $\rho$2,...,$\rho$n})
{
  1. Is EU(communicate) ¿ EU (not-communicate) execute establish joint commitment protocol;
  2. establish joint-intention $[\alpha]_\Theta$;
  3. While NOT(status($[\alpha]_\Theta$, Achieved) $\bigvee$ status($[\alpha]_\Theta$, Unachievable) $\bigvee$ status($[\alpha]_\Theta$, Irrelevant)) Do
     {
```

```
    (a) if (satisfies (Achievement-conditions($\alpha$), f) $\bigvee$ satisfies (Unachievability-conditions($\alpha$),
        f) $\bigvee$ satisfies (Irrelevance-conditions($\alpha$), f))
        /* This is the case where fact f is found to satisfy the termination condition of $\alpha$. */

        {
          i. if EU(communicate) ¿ EU(not-communicate) propose-reactive-plan Communicate
             (terminate-jpg($\alpha$), f, $\Theta$) with high priority;

          ii. if no other higher priority reactive-plan, in parallel
             Execute-individual-reactive-plan(Communicate(terminate-jpg($\alpha$), f, $\Theta$), self, $\alpha$/C,
             {$\rho$1, $\rho$2,...});
          iii. Update-state (team-state($\Theta$), f);
          iv. Update-status($[\alpha]_\Theta$);
        }
    (b) if agent-status-change($\mu$), where $\mu \in \Theta$
        {
          i. Evaluate role-monitoring constraints;
          ii. if role-monitoring constraint failure cf such that (satisfies (Unachievability-condi-
             tions($\alpha$), cf) then update-status($[\alpha]_\Theta$);

        }
    (c) if receive communication of terminate-jpg($\alpha$) and fact f
        {
        if (satisfies (Achievement-conditions($\alpha$), f) $\bigvee$ satisfies (Unachievability-conditions($\alpha$),
        f) $\bigvee$ satisfies (Irrelevance-conditions($\alpha$), f))
        {
          i. Update-state (team-state($\Theta$), f);
          ii. Update-status($[\alpha]_\Theta$);
        }
        }
    (d) Update-state(team-state($\Theta$), actions($\alpha$));
        /* execute domain-specific actions to modify team state of $\Theta$ */
    (e) if children reactive-plan $\beta$1,$\beta$2,...$\beta$n of $\alpha$ proposed as candidates
        {
          i. $\beta$i $\leftarrow$ select-best{$\beta$1...$\beta$n};
          ii. if (teamtype(Agent($\beta$i)) $\bigwedge$ ($\Theta$ = Agent($\beta$i))) then in parallel
             Execute-team-reactive-plan($\beta$i, $\Theta$, $\alpha$/C, {$\rho$1,$\rho$2...});
          iii. if (teamtype(Agent($\beta$i)) $\bigwedge$ (Agent($\beta$i))$\subset$ $\Theta$) then in parallel
             {
               A. Execute-team-reactive-plan($\beta$i, Agent($\beta$i), $\alpha$/C, {$\rho$1,$\rho$2...});
               B. Instantiate role-monitoring constraints;
             }
          iv. if self(Agent($\beta$i)) then in parallel
             {
               A. Execute-individual-reactive-plan($\beta$i, self, $\alpha$/C, $\rho$1...);
               B. Instantiate role-monitoring constraints;
             }
        }
    } /* End while statement in 4 */
4. terminate joint intention $[\alpha]_\Theta$;
```

```
5. if status([$\alpha$]$_\Theta$, Unachievable)
   {
   if ($\alpha$ != Repair) /* If $\alpha$ is not itself Repair */
   {
   Execute-team-reactive-plan(Repair, $\Theta$, C, {$\alpha$, cause-of-unachievability,...})
   /* Repair enables recovery by substitution of another team member, for instance. Cause-of-
   unachievability, passed as a parameter to Repair, may be role-monitoring constraint violation
   as in case 4b, or the domain-specific unachievability conditions. */
   } else {
   Execute-team-reactive-plan(Complete-Failure, $\Theta$, C, {$\alpha$, cause-of-unachievability,...})
   /* If Repair is itself unachievable, complete-failure results */
   }
   }

  } /* end procedure execute-team-reactive-plan */
```

Appendix B: STEAM Sample Rules

The sample rules described below follow the description of STEAM provided in this article, and essentially help encode the algorithm described in Appendix A. The rules, as with the algorithm in Appendix A, are based on execution of hierarchical reactive-plans, or reactive plans. While the sample rules below are described in simplified if-then form, the actual rules are encoded in Soar.

```
SAMPLE:RULE:CREATE-COMMUNICATIVE-GOAL-ON-ACHIEVED
/* This rule focuses on generating a communicative goal
if an agent's private state contains a belief that satisfies
the achievement condition of a team reactive-plan [OP]$_\Theta$. */
IF
agent $\nu$i's private state contains a fact F
AND
fact F matches an achievement condition AC
of a team reactive-plan [OP]$_\Theta$
AND
fact F is not currently mutually believed
AND
a communicative goal for F is not already generated
THEN
create possible communicative goal CG to communicate fact F to team
$\Theta$ to terminate [OP]$_\Theta$.

SAMPLE:RULE:CREATE-COMMUNICATIVE-GOAL-ON-UNACHIEVABLE
/* This rule is similar to the one above. */
IF
agent $\nu$i's private state contains a fact F
AND
fact F matches an unachievability condition UC
of a team reactive-plan [OP]$_\Theta$
AND
```

```
fact F is not currently mutually believed
AND
a communicative goal for F is not already generated
THEN
create possible communicative goal CG to communicate fact F to team
$\Theta$ to terminate $[OP]_{\Theta}$.
```

SAMPLE:RULE:DECISION-ON-COMMUNICATION

```
/* This rule makes the communication decision. */
IF
CG is a possible communicative goal to communicate fact F to team
$\Theta$ to terminate $[OP]_{\Theta}$
AND
Estimated value of non-communication for CG is medium
AND
Estimated value of communication for CG is low
THEN
post CG as a communicative goal to communicate fact F to team
$\Theta$ to terminate $[OP]_{\Theta}$
```

SAMPLE:RULE:MONITOR-UNACHIEVABILITY:AND-COMBINATION

```
/* This rule checks for unachievability of role-monitoring
constraints involving an AND-combination. */
IF
A current joint intention $[OP]_{\Theta}$ involves an AND-combination
AND
$\nu$i is a member performing role to execute sub-reactive-plan op
AND
no other member $\nu$j is also performing role to execute sub-reactive-plan op
AND
$\nu$i cannot perform role
THEN
Current joint intention $[OP]_{\Theta}$ is unachievable, due to a critical role failure
of $\nu$i in performing op
```

References

[1] Mihai Barbuceanu and Mark Fox. The architecture of an agent building shell. In M. Wooldridge, J. Muller, and M. Tambe, editors, *Intelligent Agents, Volume II: Lecture Notes in Artificial Intelligence 1037*. Springer-Verlag, Heidelberg, Germany, 1996.

[2] Cristiano Castelfranchi. Commitments: from individual intentions to groups and organizations. In *Proceedings of International Conference on Multi-Agent Systems*, pages 41–48, 1995.

[3] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.

[4] Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen, and Josh Clow. Quickset: Multimodal interaction for distributed applications. In *Proceedings of the Fifth Annual International Multimodal Conference (Multimedia '97)*, pages 31–40, 1997.

[5] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, 1994.
[6] B. Grosz. Collaborating systems. *AI magazine*, 17(2), 1996.
[7] B. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.
[8] B. J. Grosz and C. L. Sidner. Plans for discourse. In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 417–445. MIT Press, Cambridge, MA, 1990.
[9] J. Hendler and R. Metzeger. Putting it all together – the control of agent-based systems program. *IEEE Intell. Systems and Their Applications*, 14, March 1999.
[10] M. N. Huhns. Networking embedded agents. *IEEE Internet Computing*, 3:91–93, 1999.
[11] M. N. Huhns and M. P. Singh. All agents are not created equal. *IEEE Internet Comp.*, 2:94–96, 1998.
[12] Nick Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artif. Intell.*, 75, 1995.
[13] Nick Jennings. Agent-based computing: Promise and perils. In *Proceedings of the International Joint Conference on Artificial Intelligence*, August 1999.
[14] Nick Jennings, T. J. Norman, and P. Faratin. ADEPT: An agent-based approach to business process management. *ACM SIGMOD Record*, 27(4):32–39, 1998.
[15] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling Web sources for information integration. In *Proceedings of the National Conference on Artificial Intelligence*, 1998.
[16] Sanjeev Kumar, Philip R. Cohen, and Hector J. Levesque. The Adaptive Agent Architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the International COnference on MultiAgent Systems*, pages 159–166, 2000.
[17] H. J. Levesque, P. R. Cohen, and J. Nunes. On acting together. In *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, Calif.: AAAI press, 1990.
[18] David L. Martin, Adam J. Cheyer, and Douglas B. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artif. Intell.*, 13(1-2):92–128, 1999.
[19] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.
[20] Charles Rich and Candace Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the International Conference on Autonomous Agents (Agents'97)*, 1997.
[21] Munindar P. Singh. A customizable coordination service for autonomous agents. In *Proceedings of the 4th international workshop on Agent Theories, Architectures and Languages (ATAL'97)*, 1997.
[22] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11:36–46, 1996.
[23] C. Szyperski. *Component software: Beyond object-oriented programming*. Addison Wesley, Menlo Park, CA, 1999.
[24] Milind Tambe. Towards flexible teamwork. *Journal of Artif. Intell. Research*, 7:83–124, 1997.
[25] Milind Tambe, Jafar Adibi, Yaser Alonaizon, Ali Erdem, Gal Kaminka, Stacy Marsella, and Ion Muslea. Building agent teams using an explicit teamwork model and learning. *Artif. Intell.*, 110(2), 1999.
[26] Gil Tidhar. Team-oriented programming: Preliminary report. Technical Report 41, Australian Artificial Intelligence Institute, 1993.
[27] Gil Tidhar. Team-oriented programming: Social structures. Technical Report 47, Australian Artif. Intell. Inst., 1993.

[28] Gil Tidhar, Clint Heinze, and Mario Selvestrel. Flying together: Modelling air mission teams. *Journal of Applied Intelligence*, 8(3), 1998.
[29] Gil Tidhar, Anand S. Rao, and Elizabeth A. Sonenberg. Guided team selection. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.

Social Knowledge in Multi-agent Systems

Vladimír Mařík, Michal Pěchouček, Olga Štěpánková

Gerstner Laboratory for Intelligent Decision Making, Department of Cybernetics,
Czech Technical University in Prague, Technická 2, 166 72 – Prague 6, Czech Republic
{marik,pechouc,step}@labe.felk.cvut.cz

Abstract: The paper addresses the problems of efficient representation, maintenance and exploration of social knowledge enabling task decomposition, organization of negotiations, responsibility delegation and other ways of agents' social reasoning. We focus on multi-agent systems for integration of already existing software components. It is supposed that all the social knowledge is kept separated from both the problem solving knowledge and agents' specific internal intelligence and that it is organized and administered in the acquaintance models located in the agents' wrappers. A specific tri-base acquaintance model (3bA) is formalized and discussed throughout the paper. This model helps to optimize the communication traffic, to implement meta-reasoning processes and supports the machine learning activities. Several practical applications of the 3bA acquaintance model in different fields are presented and the acquired experience is discussed.

Motto: Knowledge is knowing as little as possible. *(Charles Bukowski)*

1 Introduction

The development in the recent decade has proven that the multi-agent paradigm represents a challenging framework for solving very complex tasks in a distributed way [35][37]. A multi-agent system (MAS) usually consists of a set of autonomous units capable of:

– independent operations aimed at meeting their **local goals**, and
– cooperative actions contributing jointly to the **global goal** shared across the community.

There are various types of agent-oriented applications. Throughout this paper we will refer to a specific category of multi-agent systems that are designed and developed in order to allow intelligent, flexible and robust integration of already existing software components. We will be addressing neither the issue of agents' specific internal intelligence, nor general reasoning machines, but we will be discussing agent's social reasoning, agent's integration in the community, responsibility delegation, task decomposition, and organization of negotiations. The agents' abilities to communicate, mutually coordinate their actions, cooperate and share the global goals determine the level of their integration-oriented behavior. These abilities depend mainly on the extent and quality of knowledge available to the agents. In this paper, we will have in mind just the **knowledge-centered aspects** of functional integration in the communities of agents of diverse nature.

Later, we will show the architecture of such an integration agent, but firstly, let us talk about knowledge, the agents acquire, administer, maintain, exploit and "own". Knowledge – a true piece of evidence in which the agent believes [40] – can either:

(i) guide agent's autonomous local decision making processes (aimed e.g. at providing an expertise or search in the agent's database) – this is what we call agent's **problem solving knowledge,** or

(ii) express the other agent's behavioral patterns, their capabilities, load, experiences, commitments, knowledge describing conversations or negotiation scenarios – which we will refer to later as **social knowledge**[1]**.**

Hereafter, when referring to knowledge we will primarily mean the agent's social knowledge. We will study agent's architectures enabling to represent, maintain and explore locally the social knowledge needed for efficient community activity integration from the global point of view.

Undoubtedly, the multi-agent systems should be equipped with a vast portion of knowledge to perform highly efficient cooperative behavior and to achieve global solutions. Such knowledge can be – in the extreme cases – stored either **centrally**, in a fully informed central unit, **or locally** owned by each of the agents. The latter case fits better the general visions how the multi-agents systems should be organized and implemented.

The main questions connected with the "local ownership" of the global knowledge are:

– What should be the reasonable extent of global knowledge administered locally, by individual agents?
– How much the agents should know about the global rules and knowledge ontologies?
– How much do they need to know about the particular cooperating "colleagues"?
– How to structure the locally stored knowledge to enable its efficient up-date and maintenance leading to reduction of the communication load in the multi-agent community?

Classical architecture of an agent separates his functional **body** containing the agent's individual abilities from his **wrapper** that accounts for the inter-agent communication. In our integration domain we will place in the body the encapsulated software application, problem solving knowledge and specific inference algorithms. The body is assumed to have no awareness of the multi-agent community. The **wrapper** will contain knowledge structures and reasoning mechanisms required for communication, coordination, cooperation and integration with the rest of the community. We further decompose the wrapper into two organizational layers (see Fig. 1):

[1] We will understand agent's knowledge about his own behavior, status and commitments, which is usually referred to as agent's **self-knowledge**, as a special instance of social knowledge. It is very often the case that the agent's self-knowledge is accessible to other agents in the community and it is a part of their social knowledge.

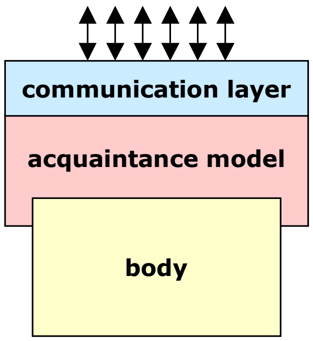


Figure 1 – An agent's body/wrapper architecture

– **communication** layer responsible for carrying out the required communication, it takes care for sending and receiving messages in a pre-determined format,
– **acquaintance** layer containing social knowledge and reasoning mechanisms needed for planning and organizing intelligent communication scenarios.

1.1 Acquaintance Model

Well organized, highly modular knowledge structures to represent social knowledge in the agents' wrappers have been introduced e.g. by [3] in the form of the model called **twin-base**. The twin-base model has been aimed at representing mainly the knowledge about capabilities, load, efficiency, reliability etc. of the cooperating agents. Models containing social knowledge are usually called the **acquaintance models.** They do represent an important category of instruments for efficient knowledge representation that supports the appropriate knowledge distribution across the multi-agent community.

An acquaintance model is a knowledge-based model of agent's mutual awareness that collects agent's knowledge about his collaborators and about suitable communication and negotiation scenarios. In the simplest possible form we can regard a *white-page* list as an instance of an acquaintance model. However we require some additional functionalities: an acquaintance model is supposed to maintain permanent, semi-permanent and non-permanent information about other agent's services, knowledge, statuses, about potential negotiation scenarios, delegation principles etc. It is required that the acquaintance model will also contain certain knowledge about his own knowledge, status and intended activities. The corresponding part of this knowledge structure can be accessible to the collaborating agents and they maintain it in an identical form.

In our understanding, the agent's body is not necessarily aware of the actual content of the acquaintance model: It is the acquaintance model as a part of the wrapper, which is responsible for exploring the social kind of information for delegating responsibilities and reasoning about other agents, etc. The real-life implementations of multi-agent systems usually don't sharply distinguish where the social knowledge is located. Let's consider all the social knowledge located outside of the body, more precisely let the acquaintance model be a part of the wrapper. This makes sense especially in the system integration oriented multi-agent systems.

In the communication or negotiation phase which should lead to certain degree of co-ordination or co-operation we can – in principle - distinguish among three simple brokering mechanisms, by means of which an agent can find the best suitable collaborating partner for delivering a required service. The mechanism based on the acquaintance model is one of them. These three mechanisms are based on:

– **broadcasting of requests** – in this case an agent sends requests for services to all members of the community and the best collaborator is selected from the subsequent replies,
– **facilitator** – where a request for service is sent to a certain central agent (facilitator) which administers all the data about the community members (yellow pages functionality) and forwards the request to an appropriate agent,
– **acquaintance model** – where each agent maintains certain amount of social knowledge about the collaborating agents and thus he is aware of their actual capabilities – the agent to be contracted is then selected without any further communication.

While the first approach demands substantial communication traffic, the second approach lowers the amount of messages sent. On the other hand the second approach, unlike the former one, depends on a central agent and the community is therefore very fragile. By using the agents' acquaintance models we may design a compromise solution. Each agent maintains in his acquaintance model the part of the information administered by the facilitator (in the second approach) that he needs and that is accessible to him. Thus each agent takes over an appropriate part of the facilitator’s knowledge.

There are several additional questions that have to be considered when designing an acquaintance model:

– What is the optimal extent of the collaborating environment (i.e. how many and which agents from the agent’s neighborhood should be included into the model)?
– How detailed the social knowledge should be?
– How to keep separated various types of knowledge, namely *permanent* (addresses of the cooperating agents), *semi-permanent* (list of capabilities), *non-permanent* frequently changing properties (load, trust, reliability, etc.)?
– How to maintain knowledge to keep it as much up-to-date as possible?

Some of these questions are solved by introducing a specific methodology for organization and administration of agent’s mutual awareness known as the **tri-base acquaintance model (3bA)** [16]**.** This methodology extends the ideas used in the twin-base model and it basically includes:

a) representation of permanent, semi-permanent and non-permanent knowledge in separate knowledge structures,
b) an inference engine for exploring and processing the locally administered social knowledge,
c) consideration of several types of social neighborhood,
d) techniques for maintaining the knowledge up-to-date by revisions of the non-permanent part of knowledge (accomplished in the idle times of the system)

leading to significant reduction of the communication traffic among the agents in the (usually time critical) phase of creation of collaborative scenarios.

As it has been discovered later, the 3bA model offers a general framework for representing knowledge in the agents' wrappers. This framework efficiently supports

- flexible internal structuring of the community (functional differentiation of agents, their grouping, hierarchical or heterarchical structuring of the agents or their groups etc.),
- reflective (and self-reflective) ways of reasoning provided the agents contain the information about their own capabilities, properties etc. in the same form as they do for other agents,
- meta-reasoning (using meta-agents equipped with the 3bA model structures as well), which proved important e.g. for solving of reconfiguration problems,
- utilization of machine learning algorithms, aimed at improvement of the system's functionality using its past or recent experience.

The 3bA methodology can be considered as a framework or guidelines for designing multi-agent systems on general level. At least, it helps to categorize and separate knowledge of diverse nature, to define scenarios for maintaining the knowledge and keeping it up-to-date as well as to analyze, classify and modularize the processes of exploring the social knowledge. The technology of maintaining the social knowledge efficiently, without any enormous communication requirements belongs to the strongest novel features of this approach.

The 3bA methodology will be presented in more details. We will further focus our attention to the role of machine learning algorithms and reflective reasoning within the frame of this methodology. Finally, the authors' experience in application of 3bA models in different applications areas (production planning, coalition formation, diagnostics, supply chain management etc.) will be summarized.

1.2 Brief Research Review

The most important virtue of the *acquaintance model* is based on the absence of a central element (facilitator, broker etc.). If some agent in the community dies or gets overloaded, the system is expected to reorganize itself in order to solve its tasks anyway. Putting too much power to a single agent – central communication agent paradigm - makes this approach too fragile and dependent on the central agent. With acquaintance models, knowledge of the central communicating agent is distributed over the community members. A number of case specific knowledge structures and maintenance algorithms for acquaintance models were implemented in the past. Let's mention some of them:

ARCHON: Acquaintance models, as agents' views of their collaborative environment, were widely used within the framework of project ARCHON (Architecture for Cooperating Heterogeneous On-line Systems) [36, 37]. ARCHON helps designers to correctly decompose and structure components of a mutli-agent systems. In a wrapper of an agent (which is called the ARCHON Layer) three types of knowledge are stored: (i) planning and co-ordination knowledge, (ii) knowledge about agent internal state, and (iii) knowledge about collaborating agents – in the form of acquaintance models.

Both the agents' problem solving knowledge and inference mechanism are stored in the Intelligent System Layer.

COVERAGE: COVERAGE is a system for multi-agent systems verification [11]. The system is capable of detecting of anomalies that can exist in MAS (of ARCHON architecture) between their *declarative knowledge* – knowledge of the Intelligent System Layer and *cooperation knowledge* stored and manipulated within its ARCHON Layer. Though no particular mechanism for acquaintance knowledge representation was presented, specific algorithms for detecting inconsistencies within the cooperation-like knowledge have been analyzed. The system is able to identify conflicts within the set of agent's problem solving knowledge (dk anomaly), among agent's social pieces of knowledge (ck anomaly), between agent's problem solving knowledge and needed social acquaintance knowledge (cd/dk anomaly) and among acquaintance knowledge of different agents (ck/ck anomaly). Methodology for detecting inconsistencies in agent's acquaintance models offers checking security in the entire MAS.

PLEIADES: This systems represents an architecture of collaborative agents making organizational decision making over the collection of internet-based heterogeneous resources [29]. The community consists of *task-specific* agents (TA) and *information-specific* agents (IA). TA co-ordinate and schedule plans with respect to a context. They collaborate in order to resolve conflicts and integrate information. IAs gather information from databases and collaborate mutually in order to provide TAs with requested information. Correspondingly, TAs agents maintain problem solving knowledge how to perform a task as well acquaintance knowledge detailing capabilities of the other TAs and IAs. The architecture has no central planning and co-ordinating unit, hence the co-ordination and planning process is spread across the multi-agent community.

VISE: A specific *twin-base acquaintance model* was successfully used in the ViSe (Virtual Secretary) intelligent agent that assists major secretarial duties [2]. The novel idea behind this paradigm is based on correct separation of information and knowledge, which an agent maintains within its wrapper. There are two independent bases proposed in the twin-base model: (i) cooperator base with all permanent data about cooperating agents (e.g. agents' addresses, message formats, agents' capabilities etc.) and (ii) task base containing up-to-date information on possible task decomposition and delegation within cooperating agents. The main innovative idea behind the twin-base approach is rooted in *periodic revisions* of the task-base, such that it contains just the most possible up-to-date information on given task decomposition and responsibility allocation. There is a special (centrally located) super-agent called a *cooperation trader* in [2] who utilizes community idle time and updates the content of the task base. This is why the most suitable and efficient agent in the community is to be requested. As a result, the communication traffic is significantly reduced and the reactions of the system are becoming substantially faster than if the broadcasting is used. This approach reduced redundant communication among agents and through intelligent cooperation both the high performance and easy maintenance were achieved.

2 Concept of the Tri-base Acquaintance Model

Now let us introduce the tri-base acquaintance model (3bA model), its underlying knowledge structures and knowledge maintenance mechanisms. Though the 3bA model is a general acquaintance model we will show its architecture and operation on a rather simple decomposition agent. The only reasoning ability of the respective agent is to decompose (in a rational way) a *task* into a set of *subtasks* and delegate the subtasks among the collaborating agents [15,18]. All this happens in the agent's Acquaintance Models while its body remains empty.

Prior to formalizing the knowledge structures of the model let us introduce several primitives we will use throughout the course of explanation. Let Θ be a set of all agents within the community and Ψ a set of all tasks the community members are able to take responsibility for. If an agent is requested to decompose a task he shall detect the best possible collaborators (based on its knowledge of decomposition) and contract these to accomplish subparts of the original request. Hereafter we will talk about such agents. This is why we may refer to a set Ψ as a collection of all tasks the agents are able to decompose.

For each agent $A \in \Theta$ let

- $\alpha(A) \subseteq \Theta$ be an *agent's total neighborhood*, a set of agents an agent A is aware of,
- $\beta(A) \subseteq \Psi$ be the set of all tasks the agent A is able to decompose,
- $\gamma(T)$, contains all possible plans for decomposing the task $T\in \Psi$. Plan for the task T is in the form $\langle T, S, O, C\rangle$, where S is a set of subtasks, which ensure completion of the task T provided that their processing meets the precedence constraints O as well as the applicability constraints C.
- $\omega(A,T) \subseteq \gamma(T)$ contains those plans for the task T an agent A knows about (if $T \notin \beta(A)$ then $\omega(A,T) = \emptyset$).

The following sets provide time dependent information. Let

- $\varepsilon^t(A) \subseteq \alpha(A)$ be the agent's current *cooperation neighborhood*, a set of agent's A collaborators at the time instant t (of course, A is a member of $\varepsilon^t(A)$, A's own cooperation neighborhood A)
- $\tau^t(A) \subseteq \beta(A)$ contain the tasks being solved by the agent A in the time instance t,
- $\pi^t(A) \subseteq \beta(A)$ be the agent's current *problem solving neighborhood*, a collection of tasks an agent A is supposed to have pre-prepared in advance in time instance t.

2.1 Knowledge Structures of the 3bA Model

Within the tri-base model each agent maintains three knowledge bases where all the relevant information about the rest of the community is stored.

Co-operator Base (CB) – maintains permanent information on co-operating agents (i.e.: their address, communication language, and their predefined responsibility). This type of knowledge is expected to be changed rather rarely. CB(A) is then defined as

$$\mathsf{CB}(A) \overset{\text{def}}{=} \{\langle B, \mathsf{Addr}(B), \mathsf{Lang}(B), \beta(B)\rangle\}_{B\in\,\alpha(A)} \text{ where}$$

$\mathsf{Addr}(B)$ specifies the agent's address, $\mathsf{Lang}(B)$ language B communicates in, as already mentioned $\beta(B)$ is a set of tasks the agent accounts for and the set $\alpha(A)$ denotes the agent's A total neighborhood (his scope of the community).

Task Base (TB) – stores (i) in its *problem section* (PRS) general problem solving knowledge – information on possible decompositions of the tasks to be solved by the agent and (ii) in its *plan section* (PLS) it maintains the actual and most up-to-date plans on how to carry out those tasks, which are the most frequently delegated to the agent - the owner of the task base, those denoted as $\pi^t(A)$. Formal definition of the $\mathsf{TB}(A)$ is then

$$\mathrm{TB}(A) \Box \langle \mathrm{PRS}(A), \mathrm{PLS}(A)\rangle, \text{ such that}$$

$$\mathrm{PRS}(A) \Box \{\omega(T,A)\}_{T\in\beta(A)} \text{ and}$$

$$\mathsf{PLS}^t(A) \overset{\text{def}}{\equiv} \{\langle T, \langle \{\langle s, B\rangle\}_{s\in S}, O, C, \mathsf{Trust}(T)\rangle\rangle\}_{T\in\pi^t(A)},$$

where for any $\langle T, \langle \{\langle s, B\rangle\}_{s\in S}, O, C, \mathsf{Trust}(T)\rangle\rangle \in \mathsf{PLS}^t(A)$ there exist O_l, C_l such that following constraints are met $\langle T, S, O_l, C_l\rangle \in \mathsf{PRS}(A)$, $B \in \varepsilon^t(A)$, $s \in \beta(B)$ and C is a specialization of C_l reflecting the considered allocation of the tasks $s \in S$, O is refinement of O_l and both O and C are valid. Each tuple in the PRS and PLS can be interpreted as a production rule expressing a possible decomposition of a given task T, the applicability constraints representing conditions on the left-hand side of the rule.

State Base (SB) has two parts, the *agent section* (AS) and the *task section* (TS).

$$\mathsf{SB}(A) \overset{\text{def}}{\equiv} \langle \mathsf{AS}(A), \mathsf{TS}(A)\rangle$$

The agent A stores in its *agent section* (AS) all relevant information characterizing the present state of the relevant part of the system (e.g. the current load of co-operating agents). This part of the state base is updated frequently and informs the agent who is busy, who is available for collaboration and makes it possible to evaluate what conditions hold at present. A sophisticated agent can include here very complex knowledge (e.g. data concerning agent's readiness to act immediately or facts the relevant agent knows at present about its environment) including knowledge about himself. To be more precise

$$\mathsf{AS}(A) \overset{\text{def}}{\equiv} \{\langle B, \mathsf{Cap}(B), \mathsf{Load}(B), \mathsf{Trust}(B), \mathsf{PresentKnowledge}(B)\rangle\}_{B\in\varepsilon^t(A)}$$

where agent's B capability has the form of $\mathsf{Cap}(B) \equiv \{\langle T, \mathsf{Cost}(T)\rangle\}_{T\in\beta(B)}$, overall agent load is $\mathsf{Load}(B)$, trust in this information is $\mathsf{Trust}(B)$ and $\mathsf{PresentKnowldege}(B)$ summarises the remaining relevant knowledge of the agent B.

The *task section* (TS) stores information on status of tasks the agent is currently solving, i.e. $\mathsf{TS}(A)$ contains relevant information on all the tasks agent A agreed to supervise recently. This set is denoted by $\tau^t(A)$. Formally

$$\mathsf{TS}(A) \overset{\text{def}}{\equiv} \{\langle T, \mathsf{Dec}(T), \mathsf{State}(T), \mathsf{Trust}(T)\rangle\}_{T\in\tau^t(A)},$$

where decomposition $\mathsf{Dec}(T)$ is taken from the $\mathsf{PLS}^t(A)$ at the moment of contract (time t). $\mathsf{State}(T)$ partitions subtasks from $\mathsf{Dec}(T)$ into three parts: subtasks finished, actually processed, and the rest. The record is complemented with the trust value $\mathsf{Trust}(T)$ denoting trust in successful completion of the plan for the task T.

2.2 Generation of Plans

Suppose the agent A is in charge of a task T. The agent can either

(i) use an existing plan stored in the *plan section* PLS of his task base or

(ii) generate a new plan using his own knowledge and inference mechanisms.

In the latter case the agent A takes problem knowledge found in the *problem section* PRS of his task base. Here he has to find a rule to be applied – a piece of general decomposition knowledge for the considered task T in the form of tuples mentioned above. After the rule is applied, the agent obtains corresponding list of subtasks $\{T_i\}$ together with both their precedence and applicability constraints. In a single moment, there can exist several rules in PRS the conditions of which are met (the data from the agent's state and cooperator bases are used for verification of these conditions) – all these rules can be fired. A good choice has to be supported by appropriate techniques of conflict resolution, of course.

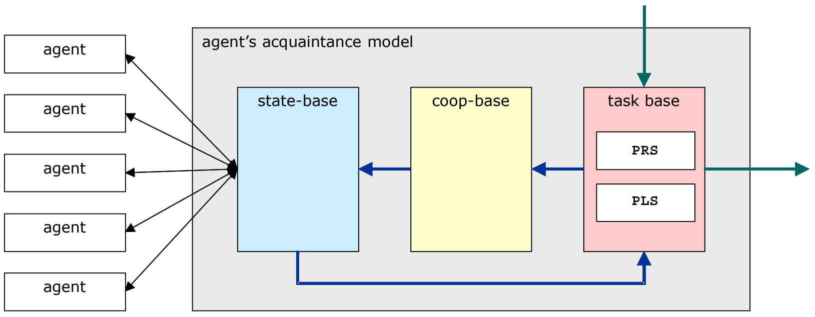


Figure 2 – Tri-base Acquaintance Model

Let us describe the process in more detail (see Fig. 2). First, the agent A consults his cooperator base in order to detect possible collaborators for all the subtasks in $\{T_i\}$. The agent A is supposed to find a good match for each task from the set $\{T_i\}$, i.e. to complement each subtask T_j with the name of an agent A_j that is ready to cope with that particular subtask. The requested result is a set of couples $\{\langle A_j, T_j\rangle\}$ meeting the following requirements:

– For each task T_j the agent A finds an agent A_j such that there is a $\langle A_j, \_, \_, \beta\rangle$[2] stored in the CB of the agent A and $T_j \in \beta$.

– For the set $\{\langle A_j, T_j\rangle\}$ the applicability conditions C are checked using information in PresentKnowledge(A) stored in the AS(A). Only those couples, for which the conditions C are met, are evaluated further.

– The evaluation of the *Trust* is carried out under consideration of the $Load_{Aj}$ and $Trust_{Aj}$ of the agents A_j as parameters, where $\langle A_j, \_, Load_{Aj}, Trust_{Aj}, \_ \rangle$ is contained in the AS(A). Those couples $\{\langle A_j, T_j\rangle\}$ together with the corresponding precedence constraints are considered and ordered according the *Trust* measure. This measure can be either minimal trust of all the agents the task is delegated to,

[2] The symbol '_' stands for an anonymous variable the value of which is not relevant

average, or weighted average. The plan with the highest *Trust* is viewed as the actual plan.

The agent section of the state base is an important resource for the plan construction. That is why whenever the contents of the AS gets updated, *Trust* is to be recomputed and each of the tuples is to be re-evaluated consequently. This kind of **re-planning** activity makes the plan the most up-to-date and bridges the planning and execution stages of the problem solving process.

2.3 Knowledge Maintenance of the 3bA Model

There are two basic techniques suitable for knowledge maintenance in the 3bA model, namely **periodic revisions** and **subscription based approach.** Let's describe briefly each of them.

Periodic revisions:

In this case, all the information contained in both the cooperator and state bases is up-dated on periodic requests. After each such up-date, the task base is up-dated consequently; that means a new ordered list of potential plans is generated. This technique was used in [3].

Subscribe/advertise approach:

This approach requires a more detailed explanation. Let us first comment how the knowledge is maintained in the **cooperator base**. As we have already mentioned, this base collects knowledge of rather permanent nature and we do not expect to update it very often besides the registering phase. Once a new agent registers with a community (by means of contacting the **facilitator** agent), facilitator replies newcomer with information about community members. In addition to this, facilitator informs other agents about the newcomer and thus invokes an update of the CB (in the form of a record append).

The **state base**, which is supposed to model an actual state of the collaborating agents, is maintained by a simple **subscribe/advertise** mechanism. After parsing the problem solving knowledge (in PRS), each agent identifies possible collaborators and subscribes them for reporting on their statuses. Let us denote the subscribing agent **subscriber** and an agent who was subscribed as a **subscribee**. Subscribee keeps advertising its load, capabilities, task completion times and costs estimates either periodically or when either of these changes. This mechanism helps the subscriber to make the best decision with no further communication.

Hereafter we will refer to a **contractor** as an agent who contracts another agent with a request. A **contractee** is an agent who was contracted by another agent with respect to a request. The **contractor** is supposed to select an optimal plan from the $\text{PLS}^t(A)$, where an appropriate amount of plans prepared in advance is stored. To do so it does not need to contract peer agents in order to find out the most appropriate (optimal) offers for further problem delegation. Knowledge stored in the PLS will help the agent to decide by itself. It is obvious that limiting the communication among agents will in its own way decrease the computational complexity of the entire problem. The price we have to pay for this, is communication increase among agents when updating the entire model. We will comment on this problem in the following paragraph.

In principle, the basic approaches to knowledge maintenance can be combined in an appropriate way. The system can be designed in such a way that e.g. whereas the subscribe/advertise mechanism is used to keep both the cooperator and state bases up-dated, periodic revisions are applied just for the task base up-date.

2.4 Saving Communication Traffic by Means of the 3bA

The prime motivation of investigating the concept of the 3bA model was saving communication traffic in a multi-agent system. Though it may look like the communication flow is not an issue, in many practical applications the amount of transmitted messages reflects the computational complexity of the problem, the system has been designed to solve.

Specific contribution to communication efficiency of the 3bA model has been tested. We have compared required communication of two types of agents – (i) a broadcasting agents that uses classical contract-net-protocol and (ii) tri-base agent that uses the collection of the social knowledge [20].

The broadcasting agent will broadcast a "tender" for collaboration. Possible collaborators subsequently reply respective collaboration offers that will reflect their capability and current status (e.g. occupation). In the last phase the broadcasting agent will select the best (e.g. cheapest) collaborator and contracts him.

The tri-base agent will analyze its acquaintance model, where he stores permanent (offered services) and non-permanent (actual status) information about all the agents that belong to his social neighborhood. According to this information, he selects the best contractor without any other communication. The price one has to pay for this saving is substantial communication required for maintenance of the acquaintance model. When mediating whether it pays off to use the tri-base approach we have to find out how often do changes of agents' statuses occur, how much of communication we will save, etc.

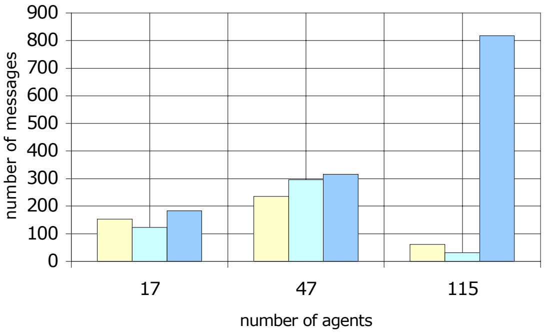


Figure 3 – Tri-base Model Savings and Maintenance

The graph in the Figure 3 shows the average communication requirements for the identical request in three different communities (the x axis specifies number of agents and the y axes draws number of messages required for solving the request). The white (left) bar shows number of messages for the tri-base agent solving the request and the gray (right) bar illustrates the broadcasting agent. The dark bar (middle) stands for messages required for the tri-base model maintenance. The communication requirements of the 3bA approach correspond to the sum of values in the left and middle column: the graph shows that tri-base reasoning brings substantial savings in multi-agent systems with complex communities.

2.5 Agents of Different Functionality

The tri-base model doesn't cover just the task decomposition and responsibility delegation tasks as the reader could understand from the previous text.. Its applicability is much wider. E.g. in the case of *configuration or quotation agents* the knowledge stored in the task base can be used to lead the communication scenarios. The body of such agents is dedicated to carry out supporting computations (of technical or economic nature). The *diagnostic agents* responsible for diagnostic data integration and fusion contain the social knowledge about sources of data (this portion of knowledge is placed in the cooperator base of the acquaintance model), about the process of finding appropriate data (task base) and about the current progress in required data processing by the other agents (state base). The data integration process itself is carried out inside the body.

If the agent is just used to carry out certain algorithms with no expected communication with the others, his task base is nearly empty. On the opposite side, some decomposition agents, which are responsible just for task decomposition could have an empty body.

3 Knowledge Improvement

Besides knowledge maintenance, which keeps the knowledge in the state base up-to-date, the 3bA concept allows knowledge to be improved as well. There are two principle ways how to implement permanent knowledge improvement:

- **Object level knowledge improvement** is based on agents' ability (and responsibility) to optimize, reorganize, deduce new pieces of knowledge and improve the knowledge stored in the acquaintance model. Object level knowledge improvement is primarily implemented **by machine learning techniques** that allow the agent mainly to find specific patterns of inter-agent communication, produce generalization, etc. Alternatively, the agent may be equipped with **meta-reasoning** (reasoning about other agents) capabilities and with explicit knowledge how the pieces of state-base knowledge may be put together and new knowledge formed.
- **Meta level knowledge improvement** is not carried out directly by the agent owning the knowledge. Improvement and knowledge revision is provided by an independent meta-agent, who observes activities of the community, collects relevant pieces of information and, consequently, tries to draw certain assumptions about the individual agent's behavior. Meta-agent can meta-reason and learn how

to enhance the community's functional efficiency and he is able to provide advice to the agents. This advice is in a form of proposed changes in the knowledge kept in 3bA models of the individual agents.

In the following we will be providing list of experiences and comments on the role of the machine learning techniques in the process of knowledge improvement. The concept of meta-agent and meta-reasoning will be presented later and we will explain how it can contribute to improving the 3bA knowledge.

3.1 Role of Machine Learning in Optimizing the 3bA Model

The results obtained using 3bA methodology are strongly determined by the content of the knowledge bases of the individual agents in the system. Any criterion designed with the aim to evaluate functionality of a multi-agent system has to reflect some observable aspects of the system's behavior. Let us consider a MAS system consisting of agents all of which are equipped by the 3bA models. Let us review those behavioral aspects we will be concerned with complemented by those parts of the 3bA model influencing strongly the corresponding type of behavior:

– The **community communication load** at a certain period of time. This value depends on the size of both the current *cooperation neighborhood* $\varepsilon^t(A)$ and the current *problem solving neighborhood* $\pi^t(A)$ of all individual agents.
– The **computation load** in critical moments and mainly the agent's ability to offer instantly a reasonable solution to a given task. This ability depends on the coverage offered by the current *problem solving neighborhood* $\pi^t(A)$ of any agent. The set $\pi^t(A)$ collects those tasks for which the "ready to use" plans have to appear in the agent's A plan section of the Task Base.
– The **quality of the plans** (corresponding e.g. to ability to accomplish the presented tasks as cheaply as possible) designed by the MAS. This feature is significantly influenced by the method used by individual agents to suggest the specific decomposition of the plan for the tasks from their *problem solving neighborhood* $\pi^t(A)$. Plans for these tasks will be pre-prepared in the agent's A plan section PLS using decompositions suggested in the problem section PRS. The problem section is treated as a sort of production system. That is why under a fixed load of all relevant agents, the choice of the most appropriate problem decomposition to be applied for a given task T depends on the *applicability constraints* of those decompositions.

None of the knowledge bases in the 3bA has to remain fixed through the life-cycle of the system. This observation holds obviously for the State Base, which is maintained in order to contain up-to-date relevant information about the cooperating agents. But the other parts of the 3bA model can be revised, too. Since our aim is to optimize the behavioral aspects mentioned above, we will search for means, which can improve behavior of the system by dynamic changes in the corresponding parts of the 3bA model. Thus we will be concerned with revisions of the *applicability constraints* of the decompositions present in the problem section PRS in the agent's Task Base as well as with revision of the *cooperation neighborhood* $\varepsilon^t(A)$ or the *problem solving neighborhood* $\pi^t(A)$. To stress the dynamic view, both the cooperation and problem-solving neighborhoods are denoted by the time tag t in the sets $\varepsilon^t(A)$ and $\pi^t(A)$ which points to the fact that the content of these sets is not permanent. We are interested in such revisions, which reflect experience the system collected during its existence – consequently we are approaching the domain of **machine learning** (ML). Are there available appropriate machine learning algorithms, which can be used to suggest adequate modifications of the upper mentioned knowledge structures describing the agent's social knowledge?

Most ML techniques have been tested by a number of authors in various MAS environments. Let us mention some of those applied recently in robotic soccer, which can be considered a characteristic example of a dynamic domain. The used techniques range from the reinforcement learning and genetic programming to decision tree learning:

– *Reinforcement learning* is used e.g. in STEAM [31] to plan how to intercept a ball or by [1] to acquire cooperative behavior in dynamic environment.
– *Genetic programming* is utilized in [13] to build agents that learn to use their basic individual skills in coordination.
– *Decision tree learning* is used in STEAM [31] for selection of an intelligent direction to shoot a ball to score a goal (using C4.5). Stone and Veloso [27] use this approach to select recipient for a pass. The confidence values from the decision tree are also used to direct the agents' actions.

A comprehensive general review of ML applications in MAS is given in [28]. This paper is not limited to applications and the used techniques, the authors stress the new challenges offered by MAS to ML. Namely they point to the fact that there is required "a method for introducing into the learning space a bias towards behaviors that are likely to interact favorably with the behaviors of other agents". In the sequel, we will show that 3bA model is well suited for that purpose. Its main advantage is transparent representation of social knowledge determining final behavior of the system. Some efficiency improvements caused by possible tri-base revisions have been studied in [20]. In the sequel we will review adaptation mechanisms and machine learning techniques as well as some minor extensions of the 3bA supporting implementation of a 3bA based multi-agent system which can learn.

Let us study those methods a **single agent** in a *multi-agent community consisting of agents equipped by 3bA models* can apply for learning how to modify his knowledge structures. The primary goal will be to design a method, one individual agent can apply on the data he has access to (most likely the record of agent's past performance) in order to revise knowledge stored in his tri-base, namely in cooperation and problem solving neighborhood and in the problem section (of his Task Base).

3.2 Changes in Cooperation Neighborhood and Its Utilization

The agent's *A* cooperation neighborhood $\varepsilon^t(A)$ contains the set of agent's *A* collaborators at the time instant *t*. More precisely, $\varepsilon^t(A)$ describes those agents, the agent *A* subscribed for reporting on their actual status. These agents are listed in the *agent section* of the agent's *State Base* [18].

Although members of the considered multi-agent community can join someone's cooperation neighborhood, so far we have not discussed means available for subscribers to restrict the cooperation neighborhood, or for a subscribee to cease advertising to a subscriber. This is a significant drawback, since subscribees do not have to be benevolent necessarily. Subscribed agents may easily interrupt advertising their status in spite of the fact they are subscribed if they have a good reason for doing so (e.g. breakdown). On the other hand, such a behavior seems to be a good reason for the subscriber to erase such "unreliable" subscribees from its cooperation neighborhood. The performative – **unsubscribe** [7] – is a natural solution to this problem as it allows subscribers to restrict their cooperation neighborhoods. This is an inverse process to that ensured by the performative **subscribe**, which expands the agent's knowledge base.

The 3bA model extended by the performative **unsubscribe** provides simple technical means for maintenance of the agents' cooperation neighborhood. When and how the cooperation neighborhood should be revised? This is the question, which has to be answered and which represents the demanding part of the considered adaptation problem.

Let us call a subscriber-subscribee link to be **unexploited** if and only if the subscriber did not contract the subscribee for certain *threshold period of time* (t_{old}). The gradual restriction of the cooperation neighborhood may be thus driven two-fold:

– either the **subscriber** agent may analyze the record of his communication history with intention to detect unexploited links (with respect to t_{old}) and unsubscribe appropriate subscribees,
– or the **subscribee** agent can analyze his record of past communication and stop advertising to agents who did not contract him for longer than t_{old} (the subscriber shall be informed about this).

If the subscriber needs to contract the previously unsubscribed agent, he subscribes him again first and then proceeds in usual decision making. The communication requirements of this process will not be worse than local broadcasting communication. Of course, the critical point of the suggested approach of **gradual revisions** is their threshold value t_{old}, which affects the overall system efficiency. That is why it has to be specified very carefully.

As soon as the agents learn to do gradual revisions of their cooperation neighborhood they can tune its size and content according to the needs of the target application domain. This task can be approached from two extreme sides:

– either the community starts with the biggest possible cooperation neighborhood, which is gradually restricted throughout the life of the community, or
– an alternative way is to start with all links unsubscribed and subscribe a new link whenever a requirement to contract arises.

The choice of the appropriate approach has to take into account the properties of the final application domain.

Another chance for improvement is to use the existing cooperation neighborhood more efficiently. It is often the case that a single agent keeps sending almost the same data or he advertises lot of unneeded information though just information about one particular resource is required. The second redundancy can be eliminated if an agent can *subscribe/unsubscribe with respect to a particular task*. To achieve so, an agent can specify the conditions (e.g. the specific task) under which he subscribes/ unsubscribes another agent. Similar mechanism can be used with advantage for optimization of subscribed information the agents advertise.

The amount of advertised messages can be also pruned with respect to *relevance of the reported update*. Let us assume that the state change of the agent A is relevant if either his load changes so that

$$100 \times \frac{|\,\mathrm{Load}_1(A) - \mathrm{Load}_2(A)\,|}{\mathrm{Load}_1(A)} \geq \xi$$

or there is at least one $\langle T, \mathrm{Cost}(T)\rangle \in \mathrm{Cap}(B)$ for which holds

$$100 \times \frac{|\,\mathrm{Cost}_1(A) - \mathrm{Cost}_2(A)\,|}{\mathrm{Cost}_1(A)} \geq \xi \,.$$

In other words we say that any advertise update is relevant if at least a single piece of advertised information changes by more than ξ percent.

A very important issue is the relation between the frequency of requests to the system and that of the agents' states updates. If the frequency of requests is much smaller than that of the agents' states updates, there is a large number of advertise messages that no one actually needs. Under such conditions it is worth considering replacing the subscription mechanism by **periodic revisions** [2][18]. So instead of the state-update back propagation in the moment of the slightest, unimportant change of a single agent within community, agents advertise in certain periodic moments. The appropriate period has to be chosen with respect to the properties of the application domain. Rare periodic revisions save lot of communications while they decrease average precision of the agents' social model. This may cause undesirable excessive communication requirements in the moment of request. Growing frequency of periodic revisions improves this precision but it increases the communication requirements simultaneously. Finally they can get close to those of the classical subscribe-advertise mechanism.

All the upper mentioned adaptation mechanisms will positively influence behavior of the considered system only if the threshold values are very well tuned to the problem solving environment. **Reinforcement learning** seems to be best suited to support their choice.

3.3 Problem Solving Neighborhood Optimization

While in the previous paragraph we have discussed optimization of subscription links and corresponding communication flows, here we will briefly comment on optimization of problem solving resources of an individual agent. The main subject of our discussion will be the amount of plans stored in the plan section (PLS) of the agent's task-base (TB) and the question how often this specific structure has to be revised.

According to [20] each change of the agents state-base (SB) will inevitably trigger a revision of the PLS since a change in the load of the subscribed agent affects all plans in PLS that rely upon this subscribed agent. These plans have to be revised and the entire structure reordered. Consequently, restriction of agent's cooperation neighborhood results in reduction of the number of events that cause PLS revision. Thus the techniques mentioned in the former section do influence this part, too.

The revision of the endangered plans (PLS maintenance) can be rather demanding, its computational complexity depends on the properties of the application domain. If the sets of agents cooperating on the plans in PLS have rather restricted intersections (plans are implemented by different agents mostly) any PLS revision is fast. However in many cases the state space modeled by the multi-agent system may be complicated and exhaustive. Some agents may administer substantial number of plans in their plan section PLS and consequently the plan revision and PLS reordering within these agents may become the limiting point of the community operation.

The simplest solution for problem solving neighborhood $\pi^t(A)$ optimization is to provide the **single agent** with the ability to identify those plans, which are obsolete or not exploited and retract these from PLS as well as restrict $\pi^t(A)$ correspondingly. There can be used similar techniques as those mentioned in the case of cooperation neighborhood optimization. Single agent can parse his history record of PLS utilization and retract *unexploited plans* (with respect to some t_{out}). Similarly, modification of the PLS is not required if none of the revised plans (or its part) changed the value of its evaluation criterion (e.g. its cost) by more than ξ in percent.

How can the optimal content of the problem solving neighborhood $\pi^t(A)$ be found? The nature of the problem indicates that for very complex domains, where some optimization of the problem solving neighborhood is inevitable, it is better to start with minimal neighborhood and build it up by experience rather than pruning the maximal neighborhood. In the latter case it may easily happen that agents will be so heavily overloaded from the very start that they will never make it up to some optimal problem solving neighborhood.

3.4 Problem Section Modification and Sub-task Assignment Optimization

Let us consider a **multi-agent production planning system**. Its main goal of is to design a plan, which ensures production of a requested commodity by efficient utilization of the available resources. The extreme types of agents in the production planning systems are the decomposition agents and the resource agents. The *decomposition agent* (DA) is responsible for decomposition of customer's orders into a sequence of simpler tasks while he has no production abilities of his own. The *resource agent* (RA) on the other hand is able to ensure those most elementary steps of the production, which cannot be further decomposed. How demanding is it to reach the planning goal for a group of agents? Critical information from the point of view of communication load is the size of the problem section PRS of the task-base. Is the optimal functioning of the whole system ensured if each of the constituting agents takes the locally optimal choice? We will show that optimal behavior of each agent with respect to sub-task decomposition does not ensure that the system will operate optimally for a sequence of requests.

Example: Let us consider a system with three agents – one decomposition agent DA and two resource agents RAs. The DA is able to decompose tasks x and y. To accomplish the task x, two elementary production steps a and b have to be done in such a way that a has to be finished before b starts, this is denoted by ($x = \langle a < b \rangle$). The only production step corresponding to the task y is c (denoted $y = \langle c \rangle$). Number of time units needed by each of the RAs to perform the considered tasks is given in Table 1 ("no" means that the agent cannot perform that task).

agent\task	a	B	c
RA_1	4	3	no
RA_2	no	2	6

Table 1 – Resource agents' services

Let DA be required to plan x and y (the requests for x and comes arrive in this order in a short time span, nearly simultaneously). If DA plans under assumption of local optimality the plan is RA_1: $a_{(0–4)}$ and RA_2: $b_{(4–6)}$, $c_{(6–12)}$ and the total execution time is 12. But the globally optimal plan would be RA_1: $a_{(0–4)}$, $b_{(4–7)}$, and RA_2: $c_{(0–6)}$ with the total execution time 7 only. In the first plan (see Fig. 4) the agent RA_2 becomes a **bottleneck** for the problems actually solved by the decomposition agent, while no bottleneck appears in the optimal plan.

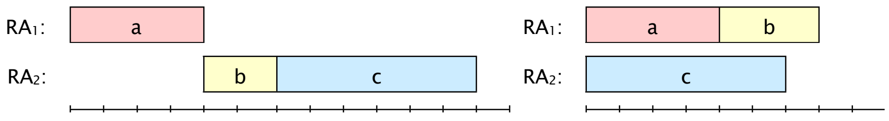


Figure 4 – Examples of locally optimal and globally optimal plan

If the analysis of past performance of the system proves e.g. that "most often the tasks x and y are requested jointly or soon after any task x there comes task y", we want the system to be able to use this information in order to avoid the local lavishness shown on the upper example. In our case, we have to specify means how to force DA under these conditions to assign subtask b to RA_1 though RA_2 seems locally better. This can be achieved by the modification within the problem section PRS of the DA agent's Task Base, namely by *change of the applicability constraints* of the corresponding plan. The language describing the applicability constraints has to provide the means to express occurrence of such a situation. Since ordering of events plays an important role here it is obvious that this language has to contain predicates and logical connectives. Moreover, some modalities (e.g. often, soon-after) should be included as well. To find out those important properties of the domain, which significantly influence functionality of the corresponding multi-agent system and the corresponding minimal language for describing these properties in the form of applicability constraints in PRS, is an interesting research challenge. It should be noted here that this is the same language as that used to describe the agent's present knowledge in the AS(A).

The task of problem section modification itself has to be divided into several parts. We will briefly mention just three of them:

(i) The first learning issue is to predict future events or to find regularities in the behavior of the problem domain (e.g. "soon after x there comes task y").

(ii) But even if the sequence of future requests is known, the algorithm to find the optimal assignment is very complicated and time demanding. Thus the next learning goal is to find appropriate "shortcuts" (alternative simple rules), which can substitute for this complicated algorithm and lead most often to the same (optimal) result, but in more efficient way.

(iii) Finally, we have to learn the applicability constraints of the simplified "shortcut" algorithm for sub-task assignment designed in the step 2. – they allow to distinguish when to use the simplified algorithm and when to use the sophisticated one.

The first experiments concerning the second and third parts have been performed for a rather simple domain [42] with intention to learn the constraints (see Part 3). Four ML algorithms have been tested, namely C4.5, Naive Bayes, linear regression and Decision Table. The Decision Tree/Table algorithms gave significantly better results for the considered purpose.

Up to now, we have been discussing learning, which is performed in a distributed way within a single agent who can apply the learned knowledge on his own. But sometimes identification of the source of inefficiency is outside of scope of a single agent. Moreover, sophisticated machine learning methods have to be applied e.g. for identification of regularities in the appearance of the tasks. To equip all agents with such complex methods can be classified as profligacy. That is why it seems advantageous to view the tasks of learning from a global perspective – such an overview gets the meta-agent who observes activities of all the agents.

4 Meta-agents and Reflectivity in Multi-agent System

As stated previously agents may collect in their acquaintance models knowledge about other members of the multi-agent community and about themselves as well. Let us talk about specific type of agents – the meta-agents – and let us discuss how they acquire, maintain and exploit knowledge but most importantly how they revise agents' acquaintance models.

4.1 Reflective Reasoning and Meta-reasoning

Prior to giving a clear-cut explanation about what a meta-agent is let us make clear several other terms. Let's consider a computational system that is capable of certain class of decision-making; for example, production planning, weather forecasting or language translation. Such a system will carry out computation (manipulation with symbols) in order to perform behavior that will meet its designed objective. Let us regard this computation as an instantiation of the system's **primary reasoning** (object-level reasoning). If we require this computational system to be reflective, it needs to be able to reason about itself, its knowledge, problem solving strategies, scope of competence and record of past behavior. A reflective system consists of **object components** and a **reflective component**. The reflective component can have a number of levels corresponding to the intended type of knowledge. It can be equipped by deduction mechanisms as well as incorporate self-representation, reasoning mechanism about such self-representation or a mechanism for controlling interaction among the object components of the system. **Reflective reasoning** [15] allows the system to analyze and learn from its past course of decision-making (*learning*), detect inconsistencies in manipulated knowledge (*reality check*) or suggest efficiency improvements in the respective problem solving processes (*adaptation*).

While reflectivity has been studied thoroughly in the area of mathematical logic, computer science and knowledge engineering, we will investigate the concept of reflectivity in multi-agent systems. A **reflective multi-agent system** should contain either a single agent – **meta-agent** – or a collection of meta-agents who are capable of reasoning about the agent's who carry out the primary decision-making. The meta-agent is able to reason about other agents, their knowledge, about their reasoning processes. The **meta-agents** individually or collectively constitute a reflective component of the multi-agent system. The agents that are the subject of reflective reasoning (these constitute the object level component of the system and carry out primary decision making) will be termed **object-agents** (see Fig 5.).

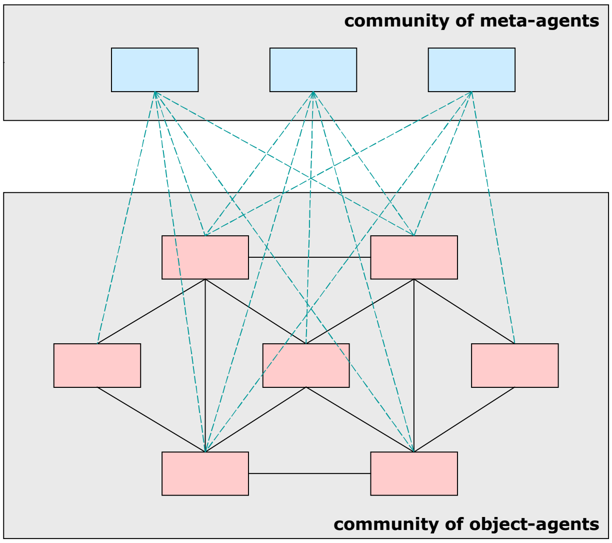


Figure 5 – meta-agents and object-agents

Two capabilities distinguish the particular type of agent that we have above called the **meta-agent**. First, such an agent can draw conclusions about the reasoning of an agent or group of agents (most often such conclusions are out of scope of any individual agent). Secondly, it can break or *reflect back* to an agent or a group of agents significant information about their state and/or behavior. Note that in this way the meta-agent can modify the contents of the agent's knowledge bases and thus influence how the agent(s) will subsequently act.

For an agent system, as noted above, when we refer to the **meta-reasoning** of an agent we mean this agent's reasoning about the knowledge and reasoning processes of other members of the multi-agent community. This can be considered as a form of social reasoning. Besides the fact the multi-agent system performs **reflective reasoning** – reasoning about himself – by meta-reasoning carried out by the meta-agents, any single agent can also perform meta-reasoning (reflective reasoning about himself) as well.

Meta-reasoning is not only useful for implementation of reflection in the multi-agent systems. If we understand meta-reasoning to be ‘reasoning about reasoning’, we recognize that it can also be used for purposes other than reflection. For example, an object-agent can meta-reason about *other* agents and their reasoning. Reasoning about other peer agents and maintaining a social model of them will bring an advantage to the respective object-level agent and may thus increase its competitiveness. However, the computational requirements needed for the model maintenance and meta-reasoning are sometimes higher than the advantage brought by doing so.

It should be noted that reflection can be implemented in to a greater or lesser extent, by requiring different levels of sophistication in the meta-agents’ meta-reasoning. Consider, for example, a community of socially knowledgeable agents who play a card game during which each agent heavily reasons about some or all of the other agents’ reasoning. If there were also (separate) meta-agents responsible only for visualizing communication exchange among the object-agents, the level of sophistication of their meta-reasoning might arguably be less than that of the object-agents'.

4.2 Types of Meta-agents

From the point of view of the meta-agents' impact on the community, we can distinguish two principal types:

– **Central meta-agent** acts as a communication/ collaboration center and may directly control the actions and knowledge of at least some of the object agents to a greater or lesser extent. Examples of such meta-agents in federated architectures include (i) **facilitators**, who are communication interfaces among collaborating agents [17], (ii) **brokers,** who are responsible for finding the best possible addressee of the transmitted message [25], (iii) **matchmakers** who also suggests cooperation patterns that may be equally used in the future [5], and (iv) **mediators,** who besides facilitating, brokering and matchmaking coordinate the agents by suggesting and promoting new cooperation patterns among them [25]. In the case these agents are tightly connected to the implementation platform, they have been classified as **middle agents** [30].

– **Independent meta-agent** is a loosely coupled meta-agent of either active or passive type. **Active meta-agent** directly affects some or all of the object agents within the community. By directly delivered messages, the meta-agent may revise the acquaintance models of the object agents. **Passive meta-agent** does not influence the community lifecycle. It just simply observes and provides the user with suitable information about how the community is evolving over time. It is up to the user to perform such a change as a feedback.

Unlike central meta-agents, the independent meta-agents are not critical to interactions among the rest of the agents. The independent meta-agent observes the behavior of the community and tries to draw some assumptions about the agents’ behavior. This type of meta-agent can use certain type of domain-relevant knowledge to help the other agents to make optimal decisions and therefore improve the performance of the multi-agent system as a whole. By introducing the concept of the meta-agent, one may get an overall view of the community's functionality. The use of multiple meta-agents avoids making the community too fragile by relying on a single independent agent, which can become a bottleneck of the process of meta-reasoning. For the rest of this paper, we will be focusing on independent meta-agents, so from henceforth when we refer to a meta-agent, we will always mean the independent meta-agent.

The observations collected by an independent meta-agent can help to spot occurrence of some specific behavior patterns, the system should consider during its decision making. Let us show the connection between the observations collected by an independent meta-agent using a simple example concerned with reduction of un-achieve messages.

The un-achieve requests appear in a *production planning multi-agent system* e.g., if a decomposition agent requests sub-tasks from different agents, and at least one of these agents replies with a sorry message. In this case the agents, which already have fixed the plans for their tasks have to undo these plans, because the whole plan cannot be achieved. Naturally un-achieve messages are not desired, since they cause additional computation and it can be quite complicated to undo a plan, which is already fixed. One of the reasons why situations with many un-achieve messages arise is the fact, that the up-date of the knowledge bases takes some time.

> **Example:** Let us consider a system with three agents - two decomposition agents DA_1, DA_2 and one resource agent RA. Each of the decomposition agents is developing his own plan denoted by $Plan_1$, $Plan_2$, respectively. Suppose both DAs are trying to send a request to a RA simultaneously, both believing that the RA is able to achieve the given subtask in time as advertised. But the RA can achieve this task only once, of course. Consequently, the request, which reaches the RA first, is answered positively, while the other one negatively. Let the first request be that of DA_1. Since the request of DA_2 failed, the $Plan_2$ cannot succeed. In order to keep actions consistent with the rest of the plan $Plan_2$ that relied upon the failed task, already contracted subtasks have to be re-planned. A task is re-planned so that it is firstly cancelled (by means of an unachieved requests) and than it is planned again (by means of an achieve requests).

If the system can learn which tasks or agents are very likely to cause appearance of such behavior, this information can be used to change the activity mode so that the relevant agent waits for the answer of these agents, before sending requests to other agents. This behavior pattern has to be identified by the meta-agent and the corresponding change has to be suggested and introduced by the meta-agent, too. On one hand, this change leads to an increase of the respond-time, because the system is not used optimally. On the other hand, it can reduce the respond-time and the number of messages by reducing the number of un-achieve requests. So this kind of meta-agent learning makes sense in those settings only, where un-achieve requests occur often.

4.3 Operation of the Meta-agent

Let's consider using meta-agents in a reflective multi-agent system for improving its primary functionality. Apart from maintaining knowledge about the object-level agents and reasoning about it, the meta-agents will have to be able to affect the operation of the object-level community [6]. Meta-agents are expected to operate in three independent phases that are to be implemented as three mutually interrelated computational threads that communicate via shared knowledge structures.

- **Knowledge acquisition and maintenance phase:** In this phase each meta-agent makes sure that his knowledge about the community is up-to date. This phase implements aspects of introspective integrity as defined in [14].
- **Knowledge analysis and inference phase:** This is a key phase of the meta-agents' operation. Here the gathered knowledge is processed and findings are formulated as needed by the particular kind of meta-agent.
- **Community revision phase:** In this phase, the community is affected by the meta-agent, by which the introspective force is implemented. The meta-reasoning agent usually affects the object-level agents by sending them control messages, knowledge improvement suggestions or he informs a user who performs appropriate action.

Meta-agents are not always so lucky to have direct access to information about the object agent status or services. Therefore it is worth to list briefly several other knowledge maintenance techniques. The knowledge maintenance process may be driven either by the object agents who are subject of the meta-reasoning process, or by the meta-agents themselves. Let us leave out of consideration the most naïve approach, when a meta-agent requests a state-base update if he needs this piece of information for his inference process.

- **Periodic revisions**: The meta-agent maintains the state-base by requesting an update periodically from the object agents which he is responsible for [3]. This approach is suitable for many applications, where the agents' status changes quite often, but where the meta-reasoning process is carried out only rarely.
- **Subscription based approach:** This technique falls into the category of maintenance that is driven by the object-agents; it has been thoroughly described in section "Knowledge Maintenance of the 3bA Model".
- **Blackboard based approaches:** This is an earlier approach to knowledge maintenance and information exchange among agents in general. Agents share a knowledge structure in which each writes his state information update. This knowledge structure is accessible to the meta-agent who uses it for the state-base update. More crudely, the state-base of the meta-agent can be the shared knowledge structure accessible by the object agents. This approach is less favored as it breaches fundamental philosophies of distributed artificial intelligence and multi-agent systems by introducing a shared, central component.
- N**on-cooperative approaches**: The meta-agent can also get his information without the object agents playing an active role in the knowledge maintenance process. For instance, the meta-agent can "sniff" the communication flow within the community and can reconstruct his state-base update according to the communicated information. This approach has proved to be suitable e.g. for passive

meta-agents that are responsible for visualizing the multi-agent community operation [18].

The knowledge the meta-agent uses for his meta-reasoning operation is stored and maintained in an acquaintance model of tri-base-like structure. The tri-base formalism is a practical and sufficient mechanism for representing meta-agents awareness of the community of object agents. Let us analyse how the tri-base model may be used for such a kind of the meta-reasoning.

The **cooperator-base** is used for storing physical addresses of the object agents, for monitoring coalition memberships but mainly for recording membership to agent's **social neighborhood** $\mu(A)$ – a collection of agents that are subject of this agent's A meta-reasoning processes. While $\mu^{+}(A)$ is a set of agents which are monitored, $\mu^{-}(A)$ lists all the agents that monitor the agent A. We can say that:

$$\forall B \in \mu-(A): A \in \mu+(B).$$

In the **state-base** section of the 3bA model let's keep:

– **ground-belief** type of knowledge that has been acquired during both the knowledge acquisition and maintenance phases by being informed directly or by monitoring of the community of the object agents and
– **assumed-belief** type of knowledge that has been induced by the meta-agent himself and has been constructed within the knowledge analysis and inference phase.

The **task-base** contains a formal model of the meta-agents reasoning activity, e.g. in the form of deduction rules providing and processing relations between the ground-belief knowledge and the assumed-belief facts. This activity can be supported by machine-learning techniques (see the section "Knowledge Improvement"). The rules in the task base are often supported by (embedded) applications stored – from the point of the introduced methodology - in the agent's body.

There are two types of assumed-belief records in the state-base. These are either **hypotheses** that the meta-agent constructed and may use for further inference process or **findings** about which he wants to inform some members of the community. Once such a piece of knowledge is found the community revision phase is triggered and the respective knowledge is directly transmitted within the community in order to revise agents' 3bA models. A number of potential revisions have been categorized in [26], e.g. agent's termination, agent's creation, agent's loss of capability, agent's acquiring of new capability etc. Special revisions are considered for revising agents' bases with respect to changing agent's properties, changing the trust values or re-specifying constraints for task specific decompositions.

The idea of meta-reasoning extends the concept of the 3bA model twofold: Firstly, a mechanism for social knowledge revision, that is based on agent's self-reasoning capability, has been provided. An agent can update a record in his state-base not only by being told so, or by observing other agent behavior. He can also manipulate the already collected ground-belief and use special reasoning mechanisms that are based either on machine learning techniques or exploitation of explicit deduction knowledge in order to form a new piece of social knowledge – assumed belief. This will make the agent not only aware of what is going on in the community but it will allow him to reason about other agents, to analyze their behavior and to predict future course of actions. This capability will upgrade agents' social intelligence.

Second improvement is of a practical nature. Sometimes it is impossible to detect interesting patterns of community interactions from the single agent's point of view. This is true mainly due to the fact that the agents have usually their organizational roles and cannot monitor/understand the whole of community.

5 Applications

Let us present examples from several application areas that document viability and efficiency of the proposed 3bA methodology.

5.1 Production Planning

Production planning and resource allocation is a complex industrial problem. In simple cases a computational model of the manufacturing domain may simulate variants of the alternative plans and will identify the optimal one. Multi-agent system is an example of such computational model as it can naturally represent the hierarchical structure of the modeled manufacturing enterprise.

In real applications, however, the number of agents and the organization architecture of the system is rather complicated. Implementing a production planning process requires substantial communication among simple agents. The exhaustive communication traffic reflects the computational complexity of the problem in question.

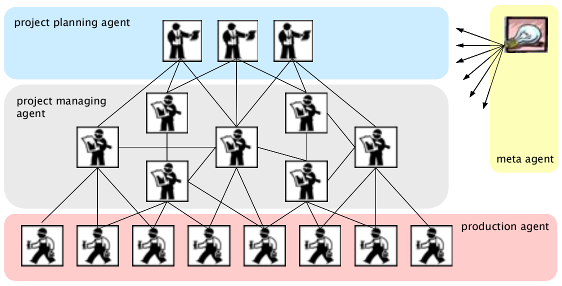


Figure 6 – ProPlanT System Architecture

Utilization of the tri-base acquaintance model limits the number of messages exchanged among the agents and therefore it cuts down complexity of the production-planning problem. System responses are fast and the substantial part of the communication is transformed to the agents' idle times. Importantly, the tri-base agents allow dynamic reconfiguration of the community. Once an agent leaves the community or changes his services, the collaborators have got the knowledge needed for efficient re-planning and only seldom they need in further communication. The hierarchy of the tri-base agents is not fixed and it allows simulation of multi-level decomposition-like decision making.

Although the agents maintain the tri-base models by themselves (using the subscribe/advertise mechanism) there are certain patterns of communication that are difficult to be detected from peer-to-peer interactions. Therefore the concept of meta-agent was used.

Within the frame of the EUREKA No.: 1439 Project a ProPlanT multi-agent system prototype for production planning was designed and implemented [19]. This prototype is aimed at planning of the TV transmitter production processes in the Czech company Tesla TV producing TV broadcasting systems. Resulting from a thorough production process analysis the authors have identified specific information units the general production process is based on. In principle we cluster agents into two fundamental super-classes: *intra-enterprise agents* (IAE) and *inter-enterprise agents* (IEE). Among IAE we can distinguish among the following basic classes of agents (see Fig. 6):

– **Production Planning Agent** (PPA) is in charge of project planning. It is supposed to construct an exhaustive, partially ordered set of tasks that need to be carried out in order to accomplish the given project. It contracts the PMA agents.
– **Production Management Agent** (PMA) is responsible for the project management in terms of contracting the best possible PA agents (in terms of operational costs, offered delivery time, and current capacity). PMA delegates his responsibility either to another PMA or it conducts work of a group of PA agents contracted for the considered task.
– **Production Agent** (PA) belongs the lowest level production units that simulate or encapsulates shop floor production processes on the IAE. The PA carries out the parallel-machinery scheduling of given tasks and manages resources allocation via special type of database agents. On the IEE level, the PA agent may encapsulate contracted suppliers offering either services or components participating in the manufacturing process. Appropriate optimization within the community will result in the cheapest (or shortest) production plan.
– **Customer Agent** (CA) is another IEE agent. In the current implementation the CA agent is the only actor that may trigger the course of production planning. It negotiates with the PPA agent in order to specify the production requirement and both the deadline and budgetary constraints.
– **Meta Agent** (MA) is a special monitoring agent who visualises information, material and work flows across the agents' community and advises on optimal system's efficiency. It shall be noted that the community of agents will survive well with no meta-agent. ‘Ordinary’ agents are able to communicate in peer-to-peer manner, but the meta-agent is able to induce specific efficiency considerations from observation of the community workflow.

As you can see, the individual agents can – despite the fact they are functionally organized hierarchically from the point of view of logical task decomposition and resource allocation – communicate in the peer-to-peer way. Thus, their grouping into three hierarchically organized levels just reflects the functional specialization of the agents.

The first version of the ProPlanT multi-agent system was implemented in C++ and the inter-agent communication was implemented via TCP/IP connections and KQML as an Agent Communication Language. Recently, the ProPlanT methodology has been exploited within the ExPlanTech EC funded IST project (IST-1999-20171). The recent FIPA-compliant version of ProPlanT has been built on top of JADE and FIPA-OS.

5.2 Supply Chain Management

For the purposes of exploration of the 3bA model in the supply chain management tasks let us distinguish among three categories of agents:

– **customer agents (CA)** which provide the global system with requests/orders,
– **decomposition agents (DA)** which are responsible for decomposition of the customer orders into a sequence of elementary steps, and
– **resource agents (RA)** representing available resources (goods, transportation means etc.).

While resource agents provide other agents with some service or commodity, the decomposition agents are responsible for optimal decomposition of a task into a set of subtasks and further delegation among collaborating agents. This classification of agents in terms of their capabilities to decompose and delegate bears a resemblance to the Pleiades architecture of collaborative agents architecture that consists of *task-specific* agents (TA) and *information-specific* agents (IA) [29].

Whereas the architecture of the 3bA models in the wrappers of the decomposition agents is very similar to that of the PPA or PMA agents applied in the production planning area, the resource agents are very similar in their nature to the PA agents. This makes the integration of the supply chain management agent community with that focused at production planning tasks very simple. In principle, when the PPA and PMA agents are directly involved in the supply chain management negotiations and the PA agents behave as being resource agents, the JIT (just-in-time) manufacturing strategy could be quite efficiently supported by this linkage. The customer can play the same role and have the same architecture in both the tasks of production planning and supply chain management.

5.3 Coalition Formation

A promising application area of the 3bA model is planning of coalition operations. Agents group themselves into teams, coalitions and alliances where they may share knowledge and goals. One of the ways, how one can plan the coalition operation is via a central planning mechanisms. However in military operations we want to avoid relying on any central element, as eventual malfunction or intrusion detection will affect operation of the targeted coalition. In peacekeeping and humanitarian operations it is also very difficult, nearly impossible to run planning centrally [33]. Non-governmental and civilian organizations are not subordinated to any supervisory body and reluctant to provide information about their status, services and resources across the community.

While a **coalition** [22] is a set of agents (an agreement) who agreed to fulfill a single, well-specified goal, an **alliance** is a collection of agents that share general humanitarian objectives and all of them agreed to form eventually coalitions. The tri-base acquaintance model as a model of agents' mutual awareness allows sharing private information within the members of a specific alliance. Each alliance member maintains a model of the other alliance members, he is able to reason about their services and to form appropriate coalitions. There is no leader in the community and any alliance member may initiate a process of coalition planning.

Agents administer three types of social knowledge:

Public Knowledge is shared within the entire multi-agent community. This type of knowledge represents the information that is freely accessible to the members of community. As public knowledge we understand e.g. agents name, type of the organization the agent represents (government, non-governmental, military, etc.), general objective of the agent's activity, country where the agent is registered, agent's human-human contact (telephone, fax number, email), the human-agent type of contact (usually http address) and finally the agent-agent type of contact (the IP address, incoming port, agent communication language – ACL, etc.)

Alliance-accessible Knowledge is shared within a specific alliance. We do not assume the knowledge to be shared within the overlapping alliances. An agent makes public some of his knowledge about himself within the alliance, whereas an agent is forbidden to make public the knowledge it has got about another agents (a number of interesting research issues may arise once this is allowed). Members of an alliance will primarily share information about free availability of their resources and respective position. This resource-oriented type of knowledge may be further distinguished as material resources, human (professional) resources and transport resources.

Private knowledge is owned and administered by the agent himself. Though the agents are expected to share private information neither within an alliance nor within a coalition, provided they find it useful, they may communicate some pieces of the private information upon a request. As an instance of private knowledge we can consider mainly specific status of the resources the agent administers. Similarly the agent will maintain both the future plans and allocation of resources as well as his past performance. An important type of private social knowledge relates agent's collaboration preferences and possible restrictions.

A slightly redefined tri-base model (see Fig. 7) has been designed for planning peacekeeping operations. The tri-base **task-base** remained unchanged and it keeps possible coalition with respect to particular tasks. The **cooperator-base** (community-belief base in the figure) stores public information about all the community members. The state base has been broken down into two bases: **social-belief-base**, where the non-permanent information about the alliance members is stored and the **self-belief-base** where the alliance accessible information of the agent himself are stored and offered to the other alliance members.

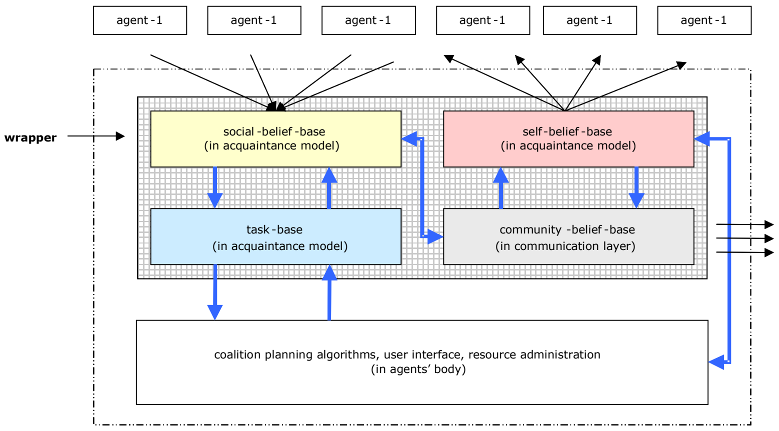


Figure 7 – Coalition Planning Acquaintance Model Architecture

Functionality of this acquaintance model, together with efficient coalition formation algorithms based on a distributed version of branch and bound algorithm has been investigated throughout the US AirForce EOARD contract no.: F61775-00-WE04. The system CPlanT developed within the frame of this project is aimed at coalition planning for humanitarian relief operations. It explores four specific classes of agents (see Fig. 8):

– **Resource agents** (R-agents) representing available supply and transportation resources,
– **In-need agents** (In-agents) representing the bodies needing in help,
– **Coordination agents** (C-agents) responsible for matchmaking the needs of In-agents with the available resources, and
– **Humanitarian agents** (H-agents) as proactive agents stimulating the matchmaking processes.

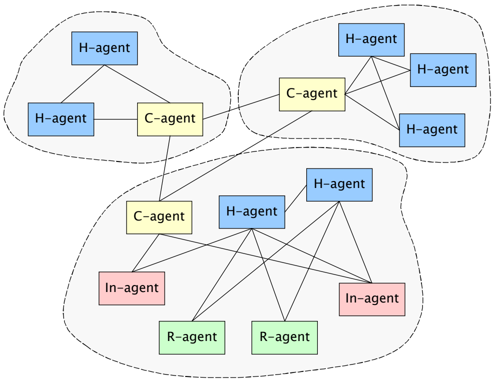


Figure 8 – CPlanT Multi-Agent Architecture

The agents are equipped by 3bA models, they are able to operate within the SufferTerra scenario as a model of a crisis situation, to create alliances and coalitions and to protect the privacy of information of individual H or R agents if required.

The overall community architecture is very flat, the C- and H-agents just mediate negotiation between the R- and In-agents and support creation of coalitions among the H- and R- agents.

The CPlanT multi-agent system has been implemented in Alegro Common Lisp Object System (CLOS). While the inter-agent communication is FIPA-compliant, each in agent CPlanT is a stand-alone application and the agents communicate via TCP/IP connection.

5.4 Intrusion Detection

The concept of acquaintance models has been investigated in the area of telecommunications to solve the problem of intrusion detection and safe communication. The agents' knowledge about the collaborators has been used for checking consistency of a possible conversation. Once there is an agent who sends a request – "requestor" to another agent – "requestee", the requestee keeps in the tri-base model a collection of specific knowledge about a requestor (or a class of requestors). He uses this knowledge in order to validate authorization of the requestor.

The problem with utilization of the acquaintance models in the area of intrusion detection is that it is rather uneasy to maintain correct knowledge about possible intruders. Therefore the concept of meta-agent is essential here. While the agents themselves are able to update basic knowledge in their models, the meta-agent has the capacity to identify suspicious patterns of communication. Moreover the meta-agent, observing the community behavior will save important amount of computational resources of the community members [20].

6 Discussion and Conclusions

Quality of coordination, cooperation and functional integration in multi-agent systems depends strongly on the **knowledge** explored. A specific part of knowledge describing behavior, availability, capabilities etc. of the agents in the community, called **social knowledge**, is crucial for efficiency of the multi-agent systems. The approach aimed at local administration of the social knowledge is strongly stressed in this paper. Here, this knowledge is strictly separated from the functional bodies of agents, quite particularly it has the form of **acquaintance models** in the agent's wrapper.

A specific **3bA acquaintance** model is presented and discussed throughout this paper. This model strictly distinguishes – similarly to ARCHON and other "classical" acquaintance models – permanent, semi-permanent and non-permanent knowledge and keeps the corresponding knowledge representation structures completely separated. To achieve an efficient knowledge maintenance, it explores the technique of periodic knowledge revisions (used originally by the ViSe system) combined with the

KQML-inspired subscribe/advertise mechanism [10]. Thus, the communication load in the critical times is strongly reduced.

A **simple inference mechanism** contained in the 3bA model explores the social knowledge represented in the form of tuples which can be interpreted as production rules. This mechanism is suitable for task decomposition, selection of cooperating agents and carrying out reasonably complicated communication scenarios. But, there is no obstacle to embody more sophisticated reasoning algorithms based e.g. on the BDI model [21] or the formal model of cooperative problem solving [38], later. The models based on the theory of joint mental states [4] providing a formal framework for the agents' joint knowledge, belief, and shared intention can take an advantage of well structured and independently maintained social knowledge that is needed for their inference processes as well. As a matter of fact, the current 3bA methodology rather **stresses simplicity and real-life implementability**. But, for majority of today’s practical industrial tasks this simplification is fully adequate and more than acceptable.

The acquaintance models in general, and the transparent structure the 3bA knowledge bases in particular, do support machine learning processes in the agents’ community. The most attractive tasks include cooperation neighborhood and problem solving neighborhood optimization as well as bottleneck detection problems. This was verified during our first experiments.

Unlike the middle-agents [5,29] who are usually a part of the multi-agent infrastructure, the tri-base acquaintance model offers a methodology for implementing agents’ wrappers that each administers an appropriate subset of the distributed yellow-page list. Each of the agents wrapped by the 3bA model plays in the community without any centrally located broker/matchmaker a role of a mediator for a certain class of tasks. However, the 3bA agents do not proactively search the multi-agent community (unlike the middle-agent, who operate mainly in the domain of information search and retrieval). The tri-base agents have been designed rather for industrial domains (planning, scheduling, manufacturing, control), where the presence of centrally located elements (agents) makes the system too fragile.

In similar domains there was also used the concept of mediators [25], which exploits primarily the contract-net-protocol negotiation schemas. The subscribe-advertise and periodic revision knowledge maintenance mechanisms in 3bA approach provide good alternative to contract-net-protocol that triggers substantial communication traffic in practical domains. Usually, there is one dynamic mediator agent cloned for a single task to be coordinated, whereas we consider the community of agents being comparatively stable (similarly as the industrial environment they represent and in which they operate). In such environments the construction or destruction of an agent appears in exceptional situations (e.g. new piece of software to be integrated, over-loaded agent, agent failing to respond, etc.).

The other aspect analyzed in this paper is the ability to achieve **reflectivity** and accomplish **meta-reasoning** using the acquaintance models. The 3bA knowledge structures and maintenance mechanisms may be used by ordinary agents as much as the **meta-agents**, who independently monitor the behavior of the community. Responsibility of classical coordination components in the multi-agent systems has been divided between the ordinary (object-level) agents and the meta-agent. Distributing the coordination knowledge among the object-level agents makes the community much robust and less sensitive to malfunctions of some key components of the systems, while introduction of the meta-agents exploring the identical 3bA methodology may provide (sometimes required) knowledge of the community as a whole. Meta-agents enable improvement of the community behavior in an evolutionary way, to react to sudden changes in the community structure (reconfiguration problems) or to changes in capabilities of individual agents, to support the process of a team or coalition formation.

A meta-agent terminology has been introduced earlier already: A meta-agent has been often assigned the role of a central communication broker [34], whom the agents communicate through. The concept of the managing meta-agent for organizing a collaborative framework is introduced in [12] and in the case of federated systems we can find the meta-agents in [8]. A similar role is played by the meta-agent in the Information Security System [9] where there is a strict hierarchical structuring of the community, with all the agents being governed and directly controlled by the meta-agent.

The notion of meta-agent as an independent observer with a qualitative reasoning unit has been introduced in [26]. Similar ideas have been exploited in the ProPlanT multi-agent system for project-driven production planning [16]. The meta-agent in ProPlanT acted independently from the operation of the community. Through observing inter-agent communication, it visualized the community architecture and communication flows, while it used simple machine learning strategies for providing various efficiency improvement suggestions [20].

There is one significant difference between a meta-agent analyzed throughout this paper as an independent observer and any kind of "central agents" like brokers, facilitators or mediators: The existence or non-existence of a meta-agent shouldn't be life-critical for the community. The community should be, in principle, capable to operate without any meta-agent, but the knowledge applied (and, consequently the overall community performance achieved) cannot be improved.

We have shown the **applicability** of this knowledge representation and maintenance technology **in several areas**, e.g. in the field of production planning and scheduling (system ProPlanT), supply chain management (project ExPlanTech), coalition formation (system CPlanT), intrusion detection in distributed information systems etc. In addition, we have carried out similar experiments with MAS equipped with the 3bA models in the field of systems diagnostics where individual diagnostic modules (sensors of diverse nature, evaluation modules based on different soft-computing techniques, etc.) have been integrated. Similarly, we have started to apply the approach in the area of distributed industrial control.

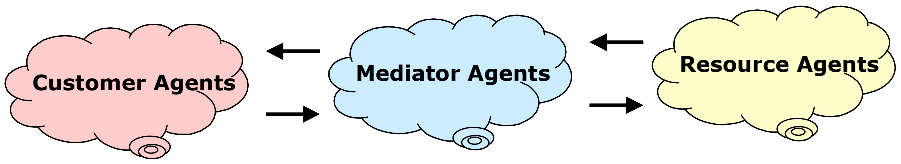


Figure 9 – Communication Flow in Multi-Agent Systems

The analysis of practical applications in different fields shows that various global community functional architectures can be constructed by means of the 3bA acquaintance model, e.g. the three-level hierarchical architecture has been used in production planning, totally flat architecture in the supply chain management area or heterarchic architecture when e.g. supply chain management being directly linked to the production planning of the supplier.

In many application areas we can see that the agents form groups according their **social roles** in the community. E.g., there is a group of customer agents negotiating and communicating with the group of resource agents through the group of mediating agents (which are called PMA and PPA agents in ProPlanT, decomposition agents in ExplanTech and coordination and humanitarian agents in the field of coalition formation) see Fig.7. Existence of these three groups of agents and their comparatively "flat" and simple linkages are typical for many areas. These observations do not exclude other possibilities to structure the agents within a group in a much more complicated way (e.g. the internal hierarchical structuring of the PMA and PPA agents in ProPlanT system).

The 3bA model should be considered as a very general acquaintance model for administrating the social knowledge. The generality of the approach allows for specific refinements of the general structures as we have documented e.g. in the case of the coalition formation problem (see Fig.5). Such specific modifications for specific applications represent the further trend.

Acknowledgements

This research was supported by the EOARD U.S.AirForce Research contracts No. F61775-99-WE099 and F61775-00-WE043 and the Grant No. MSMT 212300013 of the Ministry of Education, Youth and Sports of the Czech Republic.

References

1 Asada, M. , Uchibe, E., and Hosoda, K.: Coopearttive Behavior Acquisition for Mobile Robots in Dynamically Changing Real Worlds via Vision-Based Reinforcement Learning and Development. *Artificial Intelligence,* 110, 1999
2 Cao, W., Bian, C.-G., and Hartvigsen, G.: Cooperator Base and Task Base for Agent Modeling: The Virtual Secretary Approach. In: *Proceedings of AAAI-96 Workshop on Agent Modelling*, AAAI Press, 1996, 105–111.
3 Cao, W., Bian, C.-G., and Hartvigsen, G.: Achieving Efficient Cooperation in a Multi-Agent System: The Twin-Base Modelling. In: *Co-operative Information Agents* (Kandzia, P., Klusch, M. eds.), LNAI No. 1202, Springer Verlag, Heidelberg, 1997, 210–221.
4 Cohen, P. R. and Levesque H,. J.: Intetion is a Choice with Commitment. *Artificial Intelligence*, 42, 1990, 213-261.
5 Decker, K., Sycara, K., and Williamson, M.: Middle Agents for Internet. In: *Proceedings of Int. Joint Conference on Artificial Intelligence* 97, Japan, 1997
6 Dix, J., Subrahmanian, V.S., and Pick G.: Meta Agent Programs. *Journal of Logic Programming*, 46(1-2), 2000, 1-60.
7 FIPA – The Foundation for Intelligent Physical Agents – http://www.fipa.org
8 Genesereth, M.R., and Ketchpel S.P.: Software Agents. *Com. of ACM*, 37(7), 1994, 48-53.

9 Gorodetski, V.I., Popyack, L.J., Kotenko, I.V., and Skormin V.A.: Ontology-based Multi-agent Model of Information Security System. *RSFDGRC'99*, Japan, 1999.
10 Labrou Y., and Finin T.: *A Proposal for a New KQML Specification*. In: Technical Report TR CS-97-03, University of Maryland, Baltimore, 1997.
11 Lamb, N., and Preece, A.: Verification of Multi-Agent Knowledge-Based Systems. In: *Proc. ECAI-96 Workshop on Validation, Verification and Refinement of KBS*, 1996.
12 Lashkari, Y., Metral, M., and Maes P.: Collaborative Interface Agents. In: *Proc. of 12th National Conf. on AI*, AAAI Press, 1994.
13 Luke, S., Hohn, C., Farris, J., Jackson, G., and Hendler, J.: Co-evolving Soccer Softbot Team Coordination with Genetic Programming. In: *RoboCup'97: The First Robot World Cup Soccer Games and Conferences*, Springer, Heidelberg, 1997.
14 Maes, P.: *Computational Reflection*. Technical Report 87-2, Free University of Brussels, AI Lab, 1987.
15 Mařík, V., Pěchouček, M., Lažanský, J., and Roche, C.: PVS'98 Agents: Structures, Models and Production Planning Application. *Robotics and Autonomous Systems*, vol. 27, No. 1-2, Elsevier, 1999, 29–44.
16 Mařík, V., Pěchouček, M., Štěpánková, O., and Lažanský, J.: ProPlanT: Multi-Agent System for Production Planning. *Applied Artificial Intelligence*, vol. 14, No.7, 2000, 727–762.
17 McGuire, J., Kuokka, D., Weber, J., Tenebaum, J., Gruber, T., and Olsen G.: SHADE: Technology for Knowledge-Based Collaborative Engineering, *Concurrent Engineering: Research and Applications*, 1(3), 1993.
18 Pěchouček, M., Mařík, V., and Štepánková, O.: Role of Acquaintance Models in Agent-Based Production Planning Systems. In M. Klusch L. Kerschberg, editor(s), *Cooperative Information Agents IV* (Klusch, M., and Kerschberg, L., eds.), LNAI No. 1860, Heidelberg, Springer Verlag, 2000, 179–190.
19 Pěchouček, M., and Norrie, D.: Knowledge Structures for Reflective Multi-Agent Systems: On Reasoning about Other Agents. Report No. 538, *Department of Mechanical and Manufacturing Engineering*, University of Calgary, Alberta, Canada, 2000.
20 Pěchouček, M., Mařík, V., and Štěpánková, O.: Towards Reducing Communication Traffic In Multi-Agent Systems. *Journal of Applied Systems*, Cambridge International Science Publishing, vol. 2, No.1, 2001.
21 Rao, A.S., and Georgeff M.P.: Modelling Rational Agents with BDI-Architectures. In: *Proceedings in Knowledge Representation and Reasoning* (*KR&R-91*), (Fikes, R., and Sandewall, E.,eds.), Morgan Kaufman Publishers, 1994, 473-484.
22 Sandholm, T., and Lesser, V.: Coalition Formation among Bounded Rational Agents. In: *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95),* Montreal, Canada, 1995.
23 Sandholm, T. W.: Distributed Rational Decision Making. In*: Multi-Agent Systems: Modern Approach to Distributed Artificial Intelligence,* (Weiss, G., ed.), The MIT Press, Cambridge, MA, 1999, 201-258.
24 Shehory, O., and Kraus S.: Methods for Task Allocation via Agent Coalition Formation. *Artificial Intelligence Journal*, vol. 101 (1–2), 1998, 165–200.
25 Shen, W., Norrie, D., Barthes, J.A.: *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. Taylor & Francis, London, 2001.
26 Štěpánková, O., Mařík, V., and Lažanský, J.: Improving Cooperative Behavior in Multi-agent Systems. In: *Advanced IT Tools*, Chapman and Hall London, 1996, 293–302.
27 Stone, P., and Veloso, M.: Using Decision Tree Confidence Factors for Multiagent Control. In: *RoboCup'97: The First Robot World Cup Soccer Games and Conferences*, Springer, Heidelberg, 1998.
28 Stone, P., and Veloso, M.: Multiagent Systems: *A Survey from a Machine Learning Perspective*, Autonomous Robotics volume 8, number 3, July 2000

29 Sycara, K.: Intelligent Agents and Information Retrieval. Unicom Seminar on *Intelligent Agents and their Business Applications*, 8-9 November 1995, London, 143-157.

30 Sycara, K., Lu, J., Klusch, M., and Widoff, S.: Dynamic Service Matchmaking among Agents in Open Information Environments. *ACM SIGMOID Record*, vol. 28, No.1, 1999.

31 Tambe, M.: Towards Flexible Teamwork.. *Journal of Artificial Intelligence Research (JAIR)*, (7), 1997, 83–124.

32 Tambe, M. et al.: Building Agent Teams Using an Explicit Teamwork Model and Learning.. *Artificial Intelligence,* 110 (1999), 215-239.

33 Tate, A., Polyak, S., and Jarvis P.: Knowledge Acquisition for AI Planning within the O-Plan Project. In: *Proceedings of the 1st Workshop of the PLANET Knowledge Acquisition Technical Coordination Unit*, Salford University, Salford, UK, 1999.

34 Toomey, C., and Mark, W.: Satellite Image Dissemination via Software Agents. *IEEE Expert*, vol. 10 (1996), No. 5, 44-50.

35 Weiss, G. (editor): *Multi-Agent Systems: Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, MA, 1999.

36 Wittig, T. (editor): *ARCHON: An Architecture for Multi-agent System*. Ellis Horwood, Chichester, 1992.

37 Wooldridge, M., and Jennings, N. R.: Intelligent agents: Theory and Practice. *The Knowledge Engeineering Review*, 10(2), 1995, 115–152.

38 Wooldridge, M., and Jennings, N. R.: Cooperative Problem Solving. *Journal of Logic and Computation*, 9(4), 1999, 563-594.

39 Wooldridge, M.: Intelligent agents. In*: Multi-Agent Systems: Modern Approach to Distributed Artificial Intelligence,* (Weiss, G., ed.), The MIT Press, Cambridge, MA, 1999, 27–78.

40 Wooldridge, M.: *Reasoning about Rational Agents*, The MIT Press, Cambridge, MA, 2000.

41 Wong, H.C., and Sycara, K.: A Taxonomy of Middle-Agents for the Internet. In: *Proc. of ICMAS'2000*, Boston, 2000

42 Wurst, M.: *Applications of Machine Learning Methods in a Multi-Agent System.* Research Report No. 125/01, ISSN 1213-3000, Gerstner Lab, CTU, Prague, 2001.

Machine Learning and Inductive Logic Programming for Multi-agent Systems

Dimitar Kazakov and Daniel Kudenko*

Department of Computer Science
University of York, York YO10 5DD
United Kingdom
{lastname}@cs.york.ac.uk

Abstract. Learning is a crucial ability of intelligent agents. Rather than presenting a complete literature review, we focus in this paper on important issues surrounding the application of machine learning (ML) techniques to agents and multi-agent systems (MAS). In this discussion we move from disembodied ML over single-agent learning to full multi-agent learning. In the second part of the paper we focus on the application of Inductive Logic Programming, a knowledge-based ML technique, to MAS, and present an implemented framework in which multi-agent learning experiments can be carried out.

1 Introduction

In recent years, multi-agent systems (MAS) have received increasing attention in the artificial intelligence community. Research in multi-agent systems involves the investigation of autonomous, rational and flexible behavior of entities such as software programs or robots, and their interaction and coordination in such diverse areas as robotics [16], information retrieval and management [17], and simulation [9].

When designing agent systems, it is often infeasible to foresee all the potential situations an agent may encounter and specify an agent behavior optimally in advance. In order to overcome these design problems, agents have to learn from and adapt to their environment.

The task of optimally designing agents is even more complex when nature is not the only source of uncertainty, and the agent is situated in an environment that contains other agents with potentially different capabilities, goals, and beliefs. Multi-Agent Learning, i.e., the ability of the agents to learn how to cooperate and compete, becomes crucial in such domains.

In this paper we present an overview of important issues concerning the application of Machine Learning to Agents and Multi-Agent Systems that hopefully will stimulate further thoughts. While we point to selected relevant work in our discussions, it is not our goal to provide an exhaustive literature review. For a broader review of past research we refer the reader to [37].

* In alphabetical order

In the first part of the paper we discuss ML and MAS issues from a general viewpoint. In the second part we focus on Inductive Logic Programming (ILP), a logic-based learning technique that is able to exploit domain knowledge.

The paper is organized as follows: We begin with a general overview of the state-of-the-art in research in Machine Learning (ML) which has focused mainly on learning as an independent reasoning process and discuss the integration of these methods in a complete and situated agent system. In Section 4 we highlight several issues that are important for ML in a Multi-Agent setting, such as agent heterogeneity, awareness of other agents, communication, and distribution of learning tasks. Following this, we present a number of issues arising from the application of ILP to Agents and Multi-Agent systems and an example of such an application. We finish with a summary and an outlook on interesting open research questions within the area of Multi-Agent learning.

2 Principles of Machine Learning

In this section we briefly introduce the research area of machine learning (ML), and present some classifications of ML algorithms.

In his recent book [23], T. Mitchell defines machine learning as

> the study of computer programs that improve through experience their performance at a certain class of tasks, as measured by a given performance measure.

In more detail,

> the learning system aims at determining a description of a given concept from a set of concept examples provided by the teacher and from the background knowledge [21].

Here, 'concept' is usually interpreted as a subset of objects or observations defined over a larger set (*universe*), or, alternatively, as a boolean-valued function defined over this larger set [23]. Background knowledge is a set of concepts, i.e., statements which are known to be true for the given universe.

If the *target concept*, i.e. the concept to be learned, has a finite number of members, all of which are supplied to the learning algorithm, then the learning task can be reduced to the memorising of all examples of the concept. In this case, learning can aim at the more concise representation of the concept. However, much more often only some of the examples, called *training examples*, are used for learning. Learning from an incomplete set of examples is called *inductive learning.* Inductive learning is based on the following assumption, known as *inductive learning hypothesis:*

> Any hypothesis found to approximate the target [concept] well over a sufficiently large set of training examples will also approximate the target [concept] well over other unobserved examples [23].

The concept definition generated by inductive learning is evaluated on a set of *test examples.* None of the training examples should belong to the test set.

There are two general categories of ML algorithms: *black-box* methods, and *white-box* or *knowledge-oriented* ones. Black-box methods create a concept representation for their use, but its interpretation by the user is unclear or impossible. In this way, even if the algorithm is able, for instance, to correctly classify the unseen instances of the learned (or *target*) concept, it is not possible to *explain* its behavior. On the other hand, *knowledge-oriented* algorithms create symbolic knowledge structures that are comprehensible [21].

According to the type of examples provided, the learning algorithm may have to learn from (1) *positive* and *negative examples* of the same concept (birds and "non-birds"), (2) examples of several concepts (birds, flowers, and bees), or (3) *positive-only examples* of the same concept without any counter-examples (Donald Duck and Scrooge McDuck are birds). If in the first two cases the aim of learning is to become able to distinguish between different concepts (or the concept and its complement), in the last case the result is a description of the concept rather than discrimination from other ones. In other words, learning from only positive examples may result in a more compact representation of the target concept, or, in some statements about the target concept, which are true, but not necessarily sufficient to determine the class membership of unseen examples. Within a Bayesian framework, it has been proven that logic programs are learnable with arbitrarily low expected error from positive examples only [27]. The upper bound for expected error of a learner which maximises the Bayes' posterior probability when learning from positive examples only is within a small additive term of one which does the same from a mixture of positive and negative examples.

All three cases enumerated above belong to the *supervised learning* paradigm, where examples are annotated with their corresponding concept. Alternatively, within *unsupervised learning* the data is provided along with the definition of a language which has to be used to represent it. No additional information is supplied. Instead, there is a *language bias* specifying a certain order of preferences on the possible representations of the data. Then, among all possible descriptions of the input data, the one which has the highest score with regard to the bias is chosen.

In general, ML algorithms have to look for a somewhat more general description of the target concept than the mere list of training examples. The exact representation of the concept learned depends on the type of language used for its description. It also depends on the bias used for learning. For instance, the representation of the concept is usually expected to be more compact than the list of examples provided. In this context, a principle known as *Occam's razor* is often quoted. The statement of William of Occam (ca. 1285–1349) *"Non sunt multiplicanda entia praeter necessitatem"*—entities are not to be multiplied beyond necessity—came to be interpreted to mean that one should prefer the simplest hypothesis that fits the data [23].

Another principle, which is widely used as a language bias in ML, is *Minimal Description Length (MDL) principle.* For a given encoding, it recommends choosing the hypothesis that minimises the sum of the description length of the hypothesis and the description length of the data given the hypothesis [23].

An important issue in machine learning is also the way in which the algorithm handles misclassified examples, or noise, in the data. Even a single example, if incorrectly classified, can make the exact learning of a concept impossible and result in a description which is incomplete (i.e., does not cover all positive examples) or inconsistent (i.e., covers some of the negative ones).

More dichotomies in the taxonomy of ML algorithms are introduced by the way in which they use examples for learning. *Batch* algorithms need to process all learning examples at once and, if more examples are available, the whole learning phase has to be repeated from scratch. *Incremental algorithms*, on the other hand, are able to consider additional examples as they come. An *eager* learning algorithm generates a hypothesis for the target predicate in advance, whereas a *lazy* learner postpones that decision until it is presented with a test example. Broadly speaking, the distinction between *eager* and *lazy* learning is related to the ability of *lazy* learners to generate local approximations of the target hypothesis w.r.t. the test example, whereas *eager* learners have to commit to a single hypothesis at training time [23].

Traditional ML separates learning from the acquisition of data as two independent processes. There is some recent work that brings these processes closer. *Active learning* [40] extends a standard supervised learner requiring expensive annotated data with a module browsing a much larger unannotated dataset and selecting for annotation the examples that are considered the most beneficial to the learning process. *Closed Loop Machine Learning* (CLML) [4] couples ML with a robot carrying out experiments. In this way the learning algorithm can suggest a hypothesis and verify it experimentally — if the hypothesis is rejected, the collected data can give rise to a new round of the same cycle.

3 Machine Learning for (Single) Agents

Most of the past ML research has been focused on non-agent-based, or "disembodied", learning algorithms, i.e., without taking into account that the learning algorithm may be embedded in an agent that is situated in an environment. An agent consists of many different interacting modules (e.g., vision, planning, etc.) and the learning module is just one of them. Therefore, learning has to support and be supported by a wide range of modules.

In this section we discuss the issues arising from integrating ML algorithms in an agent system, focusing on the following questions:

– What are the learning targets and the respective training data for agents?
– What are the applicable learning techniques?
– How can the ML system be integrated into the agent architecture?

3.1 Learning Targets and Training Data

The simplest model of an agent describes it as an entity that receives sensory input from the environment and based on this input and some internal reasoning acts in the environment. Given this model, we can define the learning target of an agent simply as a decision procedure for choosing actions. We deliberately omit the notion of "choosing *optimal* actions" in this definition, because the question of what "optimal" means depends very much on the source of the training data and the bias of the learner. We distinguish three types of data sources:

External Teacher: An external teacher provides examples of actions (or action sequences) with the corresponding classification indicating their optimality or appropriateness. This model is equivalent to *fully supervised learning.*

Environmental Feedback: While the agent acts, it receives a feedback from the environment indicating the benefit of the action(s). The feedback is usually defined in terms of the utility of the current state that the agent finds itself in. This training model corresponds to *reinforcement learning.*
It should be noted that not necessarily all states will result in feedback. This means that once some environmental feedback is received it has to be propagated to all actions that potentially contributed to it. But which actions have contributed to the current state and to what extent? Certainly, actions that contributed strongly should receive more recognition (or blame). This *credit assignment problem* is still mostly unsolved. A common technique to distribute rewards (or punishments) amongst actions is to reward (or punish) more recent actions higher using a discount factor. See Section 3.2 for more details.

Internal Agent Bias: While the agent is exploring the environment with its actions, it looks out for "useful" patterns and "interesting" properties of the environment that enable the agent to generate concepts describing the environment. Usefulness and interestingness are purely based on the agent's internal bias (e.g., Minimum Description Length, Occam's razor), and no explicit feedback is given to the agent. It is assumed that the discovered concepts will help the agent to perform future specific goals efficiently and effectively. This learning model is usually denoted by the term *unsupervised learning.*

We will focus in this paper on the first two learning models. There has been relatively little work on unsupervised learning — for more information see for example [8]. In Section 5 we will present ways how to apply a supervised learning method (namely Inductive Logic Programming) in an agent context.

In the next section we discuss reinforcement learning techniques and specifically focus on the most widely used agent learning algorithm: *Q Learning.*

3.2 Reinforcement Learning

As noted above, in reinforcement learning (RL) an agent adapts his decision process based on environmental feedback resulting from its choice of actions. In this section we briefly describe a widely used RL technique, Q learning [43], and discuss the use of other ML algorithms in a RL context. For a more complete treatment of reinforcement learning and in particular Q learning we refer the reader to [23,12].

Q Learning In Q learning, an agent's decision procedure is specified by a *policy* π that maps states into actions. The environmental feedback is defined by a *reward function* R that maps states into numerical rewards. The goal of Q learning is to compute an optimal policy π^* that maximizes the reward that an agent receives.

Let s_t be the state that the agent reaches at time point t when using policy π to guide its actions (s_0 is the initial state). The reward for using policy π in state s_t is denoted by $V^\pi(s_t)$. This reward can be measured in several ways:

Discounted Cumulative Reward: This measures the cumulative rewards that an agent will get when starting in a state s_t and following a given policy π. Future rewards are discounted according to a discount parameter $0 \leq \gamma \leq 1$. The smaller γ is, the more the agent prefers short-term over long-term rewards.

$$V^\pi(s_t) = \sum_{i=0}^{\infty} \gamma^i R(s_{t+i})$$

Finite Horizon Reward: This is a modification of the discounted cumulative reward which takes only a finite number h of future states into account. The non-discounted version is:

$$V^\pi(s_t) = \sum_{i=0}^{h} R(s_{t+i})$$

Average Reward: This function measures the average reward that an agent receives for using policy π:

$$V^\pi(s_t) = \lim_{h\to\infty} \frac{1}{h} \sum_{i=0}^{h} R(s_{t+i})$$

While all above functions may have their uses, Q learning is based on discounted cumulative reward. Note that the function V defines ways in which actions receive a portion of future rewards. In the case of discounted cumulative reward, actions receive a proportion of the reward of future actions, where the size of the proportion is defined by the discount factor γ.

The optimal policy π^* can now be formally defined as:

$$\pi^* = argmax_\pi V^\pi(s)$$

The optimal policy is learned by associating an action with the direct rewards received in the resulting state and adding an estimate of the maximal

Table 1. Q Learning Algorithm

```
PROCEDURE Q-LEARN (g: discount factor) {
   Q(s,a) = 0 for all state-action pairs (s,a);
   s = initial state;
   Repeat forever {
      Select action a and execute it;
      s' = new state;
      E = maximum of Q(s',a') over all actions a';
      Q(s,a) = R(s') + g * E;
      s = s';
   }
}
```

future reward that can be achieved. The estimate is based on the agent's past experience. Given a state s, choosing an action a will result in the following *estimated* discounted cumulative reward Q_e:

$$Q_e(s,a) = R(s') + \gamma \max_{a'} Q_e(s',a')$$

where s' is the state resulting from executing action a in state s.

Given the correct Q values, the optimal policy can be computed as follows:

$$\pi^* = argmax_a Q(s,a)$$

An agent updates its estimates of the Q function while executing actions and observing the rewards of the resulting states. Table 1 shows the update algorithm (based on [23]).

The Q learning algorithm has been proven to converge towards the correct values (and thus learn the optimal policy) under the condition that the reward values (i.e., the image of R) have an upper bound. This convergence can take many of iterations. In fact, in order to guarantee optimal results all possible state-action pairs have to be visited infinitely often. Obviously, this may lead to problems in domains with continuous actions.

One of the problems that has to be addressed is how to select actions in the main loop of the Q learning algorithm. If an agent always chooses a seemingly optimal action it might get stuck in a local optimum. Therefore it is better to choose random actions with a higher probability in the early training stages and a lower probability in the later stages when the Q value estimates have converged towards the true values.

Storing all Q values in a table may quickly lead to memory space problems, and therefore a function approximation of Q, e.g., a Neural Net [39], is often used instead. This approach may lead to a more complex update mechanism.

While Q learning is an efficient and highly suitable learning method for learning agents because of its coupling of exploration and learning, it has a number of disadvantages:

– The technique is only applicable to reactive agents (i.e., agents that base their action choice only on the current state).
– Defining a numerical reward function can be a non-trivial task for some application domains.
– As mentioned above, convergence may take a long time.
– The learning result is not transparent in the sense that no explanation is given for action preferences. Nevertheless, given a state, actions can be ranked, which may yield some limited insight into the performance of the agent.

Other ML Algorithms for Reinforcement Learning Even though Q learning is the most popular reinforcement learning technique, there is no reason why other learning algorithms such as decision tree learning or inductive logic programming (ILP) could not be used in an RL context.

Džeroski et al. [7] explore the combination of relational regression tree learning and Q learning. The result is a more expressive and more general Q function representation that may still be applicable even if the goals or the environment of the agent change.

In another approach, Dietterich and Flann [6] combine explanation-based learning (a form of learning that is focused on speeding up reasoning processes) with reinforcement learning.

In Section 5 we discuss ways of combining ILP with reinforcement learning.

3.3 Integrating Machine Learning into an Agent Architecture

As Section 2 showed, machine learning algorithms can be made proactive. CLML is probably the latest stage in the development of standard ML towards active exploration of the world. It is a very specific case of a learning agent, as learning is the sole goal of the embodied agent (robot). Denying ML that privileged status and making it just one of the ways of improving the agent's performance on other tasks brings in a whole new range of issues, which will be discussed here. In the following, a simulated system of agents will be assumed, as it is the more general case, of which embodied agents acting in the real world are just an instance.

Time Constraints on Learning Machine learning algorithms are traditionally judged by the predictive accuracy of the theories learned, while time complexity is only considered as a secondary factor, a price to pay, which, if low, adds to the attractiveness of the algorithm. Time complexity may make learning from too large a data set infeasible; however, the value of a theory learned in a minute and one learned in a day is exactly the same as long as they have the same accuracy. The situation changes when machine learning is to be used in a framework of agents. There are many other activities, such as perception, planning, and control of the effectors, which compete with learning for the resources of an agent. Here, the allocation policy chosen is, obviously, of great importance.

The time constraints imposed on learning would depend on the learning task. If eager learning is to provide a theory of past experience that is more compact and also has a better generalisation power, then all that is needed is enough time for the learning algorithm to run, ideally at a time when the agent is idle so that all computational resources can be used. If lazy learning is used along with training data (past observations) to provide an interpretation for (classification of) each test example (new observation) as they come, then the time available becomes very limited. The correct recognition of a danger has to be done before the normal functioning of the agent is compromised by that danger.

Another factor for the feasibility of machine learning in agents is whether the algorithm used has a clear halting condition or it is a representative of the so called *any-time learning* in which, in general, there is always a potential for improving the current hypothesis by running the algorithm longer. Systems with exhaustive search of the hypothesis space belong to the former class; those which only sample it (e.g., genetic algorithms) to the latter. If the agent is to operate under rigorous time constraints, e.g., in real time, worst-case time complexity analysis of the algorithm can be used to determine whether to start learning at all. For any-time learning, one has to be able to implement a decision procedure to halt the learning in order to meet hard deadlines or when its cost outweighs the benefits of improved accuracy.

In general, there is a big gap between the reaction times required in Real-Time Systems (RTS) and the time complexity of machine learning at present. The situation can be compared to the one in the area of Speech Recognition (SR) and Natural Language Processing (NLP). In both cases, one has to deal with tasks which are inherently interrelated and have the same goal. Indeed, understanding speech can be improved by the use of parsing or semantic analysis, and a learning agent can potentially outperform one that uses a hard-wired behavior. The very different hardware and application requirements (it is acceptable for several NLP tasks to be carried out off-line whereas speech recognition is essentially an on-line task) kept SR separate from the rest of NLP well into the mid-1990s, when off-the-shelf hardware and software platforms, and tailor-made applications could at last meet the requirements of both areas [41]. One can expect the same development at the intersection of machine learning and real-time systems as the latter search for greater efficiency and make a conceptual shift from dealing with hard time constraints to frameworks allowing for more flexibility, such as the imprecise model of computation and flexible real-time systems [31]. It is clear though that the process of mutual approaching of ML and RTS is at its beggining.

To make the integration of ML with agents easier, one could use simulations in which the time constraints on ML are gradually taken more and more into account. One could, for instance, discriminate between the following three cases:

– unlimited time for learning
– upper bound on time for learning
– learning in real time

Synchronisation between Agents' Actions When a simulated environment of learning agents is studied, one has to choose a method that will inform an agent about the current state of its world, prompt it to choose an action, and then carry out the action and change the environment accordingly, if the action is possible. Consider the following three methods:

One-step-at-a-time, simultaneous update of environment The agents are prompted to choose their moves one by one. After each of them has selected an action, the environment carries out all the actions simultaneously. The method has to be able to deal with conflicting actions.

One-step-at-a-time, immediate update Same as above, but each agent's action is carried out immediately upon generation. Here the order in which agents take turns will be important.

Agents as asynchronous processes Of the three, this is the best model of reality. However, its implementation requires more hardware resources, such as separate computers or processors, and more complex software implementation.

The choice of the type of time constraints and time synchronisation of the individual agents' actions is schematically represented in Table 2.

Table 2. Synchronisation × time constraints

	Unlimited time	Upper bound	Real time
1-move-per-round, batch update	Logic-based MAS for conflict simulations [18]		
1-move-per-round, immediate update	The York Multi-agent Environment [14]		
Asynchronous			Multi-agent Progol [25]

Learning and Recall Each time the sensory information about the world is updated, an agent that is able to learn has to choose whether to *recall* its existing model of the world to decide about its next action, and learn from its outcome immediately to update its current model of the world, or to postpone learning for later. If lazy learning is used, the distinction between learning and recall disappears, since a new theory with a limited coverage is built to classify the new example, and all that is carried forward is the new observation (but none of the theories). In the case of eager learning though, learning and recall are two separate processes, the coordination of which will be discussed next.

Lazy learning in its simplest forms, such as case-based reasoning, only requires to search through the already seen cases and find the closest match for the new one that is to be analysed (classified). Although suitable indexing, such as hash tables, can be used to reduce look-up times, the applicability of lazy learning is ultimately bounded by its limited generalisation power, and the fact that its time complexity increases with the number of examples. Eager learning can help to obtain theories with coverage far beyond the one guaranteed by lazy learning. For instance, Explanation-Based Learning (EBL) is potentially able to learn a theory from a single example. On the other hand, there are tasks where lazy learning guarantees the best results [5,28]. However, if recall times are an issue, eager learning can be used to compress training examples (previous observations) into a more concise theory in the hope that this will also result in faster recall.

Different ways of combining learning with recall can be summarised in the following clasification:

Parallel learning and recall at all times Learning and recall are possible within the same course of actions.

Learning and recall as separate modes (one thing at a time) The agent has two separate modes. In the first, it is active and uses its existing knowledge to choose its actions, but cannot update it; observations are collected for later processing. In the second mode, the only activity performed by the agent is learning, when it processes the examples accumulated in the active phase to incorporate them into its model of the world. The examples can then be disposed of, if learning is incremental, or they can be kept if the agent uses batch learning that starts from scratch each time it is invoked. In the context of agent learning, incremental learning seems better suited as it has lesser computational and memory requirements.

Cheap on-the-fly learning, off-line computationally expensive learning The amount of raw information that an agent collects through its sensors can be considerable, which may lead to infeasible memory requirements if separate modes for learning and recall are used. In those cases, an agent could employ some limited on-the-fly learning to reduce the strain on memory. For instance, one could reduce the dimensionality of the object language, introducing a new set of attributes if necessary, cluster examples and replace each cluster with a single representative, or ignore "uninteresting" examples altogether. Then, off-line learning can be applied at a later time as in the previous case. Here one is tempted to recall the case of human learning—altough one's knowledge is apparently updated in periods of awakeness, the process continues during sleep, and the latter phase is crucial in the acquisition of knowledge for long-term use.

4 Machine Learning for Multi-agent Systems

As one moves from the single-agent setting to an environment where many agents are acting and potentially interfering with one another, acting optimally and consequently learning how to act optimally becomes a highly complex task. When applying learning techniques to multi-agent systems there are many issues to be taken into account. In the following sections we will discuss a few questions that arise in this context and point to relevant literature:

– What impact does awareness of other agents and their behavior have?
– What is the importance of communication and how does it influence the learning process?
– How does learning influence team heterogeneity?
– How does distributed learning differ from other multi-agent learning approaches?

4.1 Awareness of Other Agents

A multi-agent system is generally defined as a collection of agents that observe and act in the same environment. It is important to stress that this does not imply *social awareness*, i.e., awareness of other agents in the environment and knowledge about their behavior.

While it may seem useful to have an increased social awareness, it is sometimes not necessary in order to achieve near-optimal performance. Steels [38] describes an application domain where the task for a group of robots is to collect rock samples and return them to a central storage facility. A near optimal solution defines a small set of simple reactive rules for the robots that do not require explicit social interactions[1].

Vidal and Durfee [42] define levels of social awareness as follows:

0-level agents: have no knowledge about other agents or their actions, and observe them only as changes in the environment.

1-level agents: recognize that there are other agents around, but have no knowledge about their behavior. These agents model other agents as 0-level agents.

2-level agents: have some knowledge about the behavior of other agents and their past observations. The model of the other agents does not include any influence of their knowledge about the agent himself, i.e. a 2-level agent models other agents at most as 1-level agents.

These definitions can be continued to higher levels *ad infinitum*, where agent A knows that agent B knows that A knows that B knows, etc.

It should be noted that 0-level agents are able to learn implicitly about other agents, as long as they are able to observe the results of the actions of other agents in the environment. Even though 0-level agents have no explicit social awareness, they will incorporate other agents behavior in their learned

[1] Admittedly there is a limited amount of implicit social interaction involved in that robots drop rock samples that are used by other robots in their decision process. Nevertheless, none of the robots needs knowledge about the actions of other robots and treats rock samples in the same way, whether they have been dropped by other robots or have been there to begin with.

hypothesis as the behavior of the environment. Therefore, multi-agent learning, in principle, already starts with 0-level agents.

Mundhe and Sen studied the performance of Q learning agents up to level two in two-agent single-stage games. Interestingly, their results show that two 1-level agents display the slowest and least effective learning, much worse than two 0-level agents. In general, the results show that a diversity of agents is beneficial and that myopic agents outperform non-myopic agents. While these results are interesting, they are restricted to a specific appliation domain, and there is need for further research into these issues in order to be able to define more general principles.

4.2 Communication and Learning

In the eyes of a linguist, communication and learning are inherently related, since the same language concepts are used in both processes. A difference is usually made between *knowledge* and *skills*: the first can be clearly formulated and conveyed by the means of language; the second can only be acquired through personal experience, and the theory learned (if there is any) cannot be put in words. Using ML terminology, one could say that learning "knowledge" as defined in this context corresponds to white-box learning, and learning skills to black-box learning. It should be noted that even the exchange of parts of Neural Networks (that are usually associated with black-box learning) would require the agents to "understand" what is transmitted and thus become an exchange of knowledge rather than skills.

It is also a well-known fact that human experts are reluctant to formulate general rules they base their decisions on. One the other hand, they find it easier to motivate their decision if a particular case is studied. A quotation of the architect Frank Lloyd Wright will help to make the point: "*an expert is someone who does not think anymore—he knows*". The above descriptions match the definitions of various forms of lazy learning.

If learning is used in a single agent, the only reason to choose a learning method is the quality of its results. However, the situation changes with the move from single-agent learning to societies of learning agents which can communicate with each other. For an agent, the cost of asking information (knowledge) from another, experienced agent is usually much lower than the cost involved in acquiring the information on its own, either by exploring the environment or by purely observing the actions of other agents.

However, the information should be expressed in a formalism (language) shared by both agents. In other words, only the results of white-box learning can be shared between agents. Consider also the case when the communication channel has a (very) limited bandwidth. Although lazy learning may still be the best for the individual, some form of eager learning must be employed if the agents are to benefit from their communication abilities, as "downloading" another agent's exaustive list of personal experiences may be infeasible.

4.3 Team Heterogeneity

In a team of agents that is solving a task with combined forces, it often seems useful for agents to specialize in different subtasks and thus share the effort in a more efficient way.

One way to achieve this kind of heterogeneity in the multi-agent system is to equip agents with different sensor and effector capabilities or behaviors and thus pre-define the roles that they are going to play in the team effort. While this method may lead to good results [32], it has a number of drawbacks. Firstly, it is not always obvious how to specify an optimal (or even useful) distribution of sensors and effectors or behaviors. Secondly, it may be quite expensive (in terms of hardware or in terms of development time) to design a system of heterogeneous agents.

An alternative is to use a team of learning agents that are homogeneous to begin with, but with time and experience will diversify and specialize. Balch [2] studied the conditions under which a team of agents based on reinforcement learning will converge towards heterogeneity. In his research, he distinguishes between three types of reward functions:

Local performance-based reinforcement: Each agent receives rewards individually when he personally achieves the task.

Global performance-based reinforcement: All agents receive a reward when one of the team members achieves the task.

Local shaped reinforcement: Each agent receives rewards continuously while he gets closer to accomplishing the task.

Balch applied the different reward functions to multi-agent reinforcement learning in the domains of multi-robot foraging, robotic soccer, and formation movements. Using *social entropy* [3] as a diversity measure, he observed the following results:

- Globally reinforced agents converge towards a heterogeneous team, while local reinforcement leads to homogeneous agents.
- Heterogeneity is not always desirable: in multi-robot foraging a locally reinforced and therefore homogeneous team outperforms a globally reinforced and therefore heterogeneous team. On the other hand, in the robotic soccer domain the results are opposite, i.e., heterogeneity yields the better performance.

Clearly, these results show that team heterogeneity is an important factor in multi-agent performance and different learning methods can yield different levels of heterogeneity. It is an interesting open research problem to gain a clearer and more detailed understanding of the relationship between agent diversity and performance and learning in general.

4.4 Distributed Learning

While much of the multi-agent learning research is concerned with the question on how to learn local hypotheses that enable agents to collaborate, compete, and/or communicate more effectively, distributed learning is concerned with applying multi-agent techniques to learn a global hypothesis given local and distributed learning agents.[2]

As an example for distributed learning, consider a robotic soccer domain, where in the absence of a global view the computation of an opposing team's strategy has to be based on local observations of the individual players. It would be infeasible to simply send all these observations to a super-agent that learns from them, and therefore the agents need to first generalize from their observations and then share the results in order to compute the global strategy of the opposing team. We are currently working on algorithms to achieve this task.

Weiss [45] distinguishes three types of multi-agent learning:

Multiplied Learning: Each agent learns independently of the other agents. While there may be interactions concerning the exchange of training data or outputs, no interference in the learning process itself takes place.

Divided Learning: The learning task is distributed amongst a team of agents. The division takes place on a functional level, i.e., agents take different roles in a team and learn them separately.

Interacting Learning: Agents interact during learning and cooperate in generating a hypothesis beyond the pure exchange of data. Weiss describes this as a "cooperated, negotiated search for the solution of a learning task".

Most multi-agent learning systems fall into the first two categories. The third category describes "true" distributed learning, and very little research exists in this area.

Provost and Hennessy [34] describe a method for parallel learning where learning agents receive different subsets of the training data, learn a local hypothesis, and generalize from this a global hypothesis. An agent A checks the validity of a learned rule by sending it to another agent B in order to compute the accuracy on B's dataset. The most valid rules form the final hypothesis.

Other approaches related to distributed learning include Weiss' ACE and AGE algorithms [44] where a group of Q learning agents jointly decide on the actions via a bidding scheme and then jointly receive reinforcement for the executed actions.

5 Inductive Logic Programming and Agents

Inductive Logic Programming (ILP) is a branch of machine learning built on three pillars: *Logic Programming* (LP) as the representation formalism for both

[2] Of course, the distributed learning process may still lead to more effective coordination, but the learning metaphor is different.

training examples and theories learned, *background knowledge* in the form of predicates, which extend the concept language, *i.e.*, the one used to express the target concept, and, finally, *induction* as a method of learning.

5.1 Logic Programming as Object and Concept Language

In ILP, both the *object language* used to represent examples of the target concept, and the *concept language* used to represent the concepts (theories) learned are based on first order logic or, more precisely, on logic programming. That means that all facts and concepts are represented as predicates of first order logic. In most ILP systems, there is one target predicate only. Also, the training examples are most often represented as ground facts, *i.e.*, the object language is restricted to propositional logic in the very same way as in propositional learners such as ID3. In other words, the training data can be represented as a single table, the name and columns of which correspond to the name and attributes of the target predicate (see Table 3).

What makes ILP different from the propositional learners is the concept language used. For that purpose, ILP uses relations expressed as Horn clauses. For instance, the concept of equality of two arguments can be expressed by the clause `equal(X,X)`; to say that Argument 1 is greater than Argument 2 one writes `greater(X,Y):- X > Y`. In propositional logic, one cannot use variables to define relations between target concept arguments, so the above examples have to be covered by explicit enumeration of all pairs of `X` and `Y`, a task which is tedious if the range of the two arguments is finite, and impossible otherwise.

ILP systems may require the use of *modes* and *types. Modes* serve to distinguish between the input arguments of the predicate, *i.e.*, those which will be instantiated in a query, and the output arguments which will be instantiated as a result of the query. The former case is usually indicated by a plus '+', and the latter by a minus '–'. Systems such as Progol allow the user to indicate that in the clauses of the target predicate to be learned an argument will be instantiated to a constant by marking that argument with the hash symbol '#'.

Types specify the range of values for each argument, and are usually defined by user-defined unary predicates; some systems provide built-in types, such as `nat`, `real`, and `any` in Progol. It is important whether these predicates can only be queried with an instantiated argument (*is* 5 *a natural number?*) or are generative, allowing for the sampling of the type range (*give me a (random) natural*

Table 3. Training data representation

Good bargain cars				ILP representation
model	**mileage**	**price**	**y/n**	`:- modeh(1,+model,+mileage,+price)?`
BMW Z3	50,000	£5000	+	`gbc(bmw_z3,50000,5000).`
Audi V8	30,000	£4000	+	`gbc(audi_v8,30000,4000).`
Fiat Uno	90,000	£3000	–	`:- gbc(fiat_uno,90000,3000).`

number). This importance is related to the type of training examples available. If both positive and negative examples of the target concept are provided, the types are only used to mark which variables in the Horn clause can be unified, restricting in this way the target concept search space. On the other hand, generative type definitions are required for positive-only learning in the ILP system Progol [27]. The reason for that requirement becomes clear after a closer look at the mechanism underlying that type of learning.

There are several situations when positive-only learning would be the most natural setting: when a child learns to speak, one could hardly imagine the parents providing examples of ill-formed sentences; another example is the case of an animal or agent learning the definition of "non-lethal action". If the search for the target concept is only guided by the accuracy as tested on the training examples, the trivial definition classifying any entry as a positive example would cover all positive examples, and no negative ones (since none are provided). The (quite ingenious!) solution provided to that problem in Progol is to generate *random examples* of the target predicate by sampling the type range of each attribute, and then treat these examples as negative. This approach proves successful when combined with a learning bias based on a trade-off between the theory size (the shorter, the better) and its coverage (the more positive and the fewer random examples covered, the better).

5.2 The Importance of Background Knowledge

In the examples of `equal/2` and `greater/2` as predicates that ILP can learn, only very simple mathematical relations such as identity and "greater than" have been used. In fact, the concept language of propositional learners can easily be extended to include these relations. The truly unique feature of ILP is that one can define background knowlegde — a set of relations (predicates) that can be used in the definition of the target concept. The use of certain background predicates may be a necessary condition for finding the target concept. On the other hand, using redundant or irrelevant background predicates slows the learning down.

For instance, one can use the predicate `imported_model/1` to search for a potential correlation between that feature and the price of a car; in case of large import taxes that predicate may prove quite useful. This is still an example where all information used in learning can be represented in a single table by adding an extra attribute to it. The target concept, in its simplest case (`imported_car=no`) is also propositional. To show the advantage of using ILP, imagine that the product of the car mileage and price was a good indicator of its being a good bargain. One could then define a background predicate `prod(Miles,Price,Threshold) :- Miles*Price < Threshold` and use it with modes `:- modeb(1,prod(+Miles,+Price,#Threshold))?` to learn the following theory from the data in Table 3:

```
gbc(Model,Miles,Price):-
      prod(Miles,Price,250000001).
```

One can also allow the target concept to be recursive when appropriate or define constraints such as "no car is a good bargain if one can buy the same model and mileage for less". The latter can be expressed in Progol as:

```
prune(gbc(Model,Miles,Price),Body) :-
      Body, gbc(Model,Miles,Price2), Price2 < Price.
```

5.3 Inductive Learning

Rather than proving clauses starting from some general theory, as other LP systems, such as Explanation-Based Learning, do [22], ILP systems try to find a general theory that can explain the given clauses. While resolution is generally used for automated deduction, there are many ways to perform induction. Some of them can be generally called *inverse resolution* [29], since they can be derived from applying the resolution step in the reverse way. There are also other induction techniques, such as least general generalisation (lgg) [33].

5.4 ILP Learning for Agents

The description so far shows ILP as symbolic, knowledge-oriented learning which can explore complex hypotheses and represent them in a concise way. Here the discussion of ILP continues with a number of implementation-specific issues, and their relevance to the use of ILP in agents.

Learning Pure Logic Programs *vs.* Decision Lists ILP algorithms can be divided into those that learn theories expressed as pure logic programs (FOIL [35], Golem [30], and Progol [26] fall into that group), and others which include extra-logical predicates. A concept language including the predicate *cut* (`!`) allows for the learning of *first order decision lists*, in which each clause (but the last) ends in a cut. In pure logic programs, the clause order is irrelevant. However, when a decision list is queried, the list is searched in a top-down fashion, and only the first applicable clause is used. The intuitive notions of rule and exception are easily represented with decision lists: if the exception preceeds the rule, the latter does not have explicitely to mention that it does not cover the former. A decision list example follows in Table 4. The example models the behavior of a cat, which is afraid of all dogs but the ones that belong to its owner, and has no fear of other animals.

Theories expressed as decision lists are often simpler then their corresponding pure LP representation. Decision lists can be useful to integrate previously learned theories with newly acquired training examples. If the examples are misclassified by the existing theory, they can be added as exceptions at the top of the decision list to ensure their correct classification. Later on, learning can be used to replace these exceptions with rules, if possible. Foidl [24] and Clog [15] are two of the first order decision list learners. It is also worth mentioning that Clog, unlike Progol, is an incremental learner.

Table 4. Example of decision lists

```
%action(Cat,ObservedAnimal,Action)
action(Cat,Animal,stay) :-
   dog(Animal),
   owner(Owner,Animal),
   owner(Owner,Cat),!.

action(Cat,Animal,run) :-
   dog(Animal),!.

action(Cat,Animal,stay).
```

Eager ILP *vs.* Analogical Prediction The relative merits of eager and lazy learning have already been discussed. ILP belongs to the eager learning paradigm, with the sole exception of Analogical Prediction (AP) [28]. AP constructs a separate theory (first-order Horn clause) to classify each test example; at the same time, that theory provides an explanation of the decision made, which is uncommon for lazy learning. AP would be suitable in the context of a rational learner that has to provide motivation for its actions. However, the time complexity of the method can present a serious obstacle to its larger use.

Single-predicate *vs.* Multi-predicate Learning When learning is judged in terms of predictive accuracy, the use of additional training examples can only be beneficial, if there is no noise in the data. However, in agent learning one should also be concerned about the time complexity of the theory learned, *i.e.*, how long it takes to query that theory. Since ILP learns logic programs, which can be recursive or allow for a lot of backtracking, the complexity of these programs can easily become an issue. The only experimental results known to the authors involve bottom-up ILP based on *lgg* [13], p. 117. In those experiments, the curve representing the relationship between the number of training examples, and the average time needed to prove a goal or reject it within the theory learned, is bell-shaped rather than linear. To avoid learning theories of high time complexity, an agent could content itself with using a simple, low-coverage theory until there are enough additional training examples to produce a theory the complexity of which is beyond the critical point.

Existing Work ILP has been applied to a number of tasks related to agents. For instance, Džeroski et al. have combined the ILP system TILDE-RT with reinforcement learning [7] for the task of learning actions in a simple world of blocks. The Q-function is learned in the form of a logical regression tree, *i.e.*, a binary tree in which each non-leaf node corresponds to a condition expressed as a logic clause, and is split into two branches labelled ‘yes’ and ‘no’, whereas leaf nodes are labelled with Q-values.

Reid and Ryan have used ILP to improve planning in hierarchical reinforcement learning , in which ‘optimality is traded for speed hoping to find good solutions quickly by breaking a monolithic reinforcement learning task into smaller ones and combining their solutions” [36]. Rather than learning a Q-tree, here ILP is used to learn descriptions for subsets of the state space.

Matsui et al. have proposed to develop an ILP agent that avoids actions which will provably fail to achieve the goal [20]. The suggested application area is the one of RoboCup, a robot football tournament with world-wide popularity.

Another ILP application in the same domain provides the agent with a way of analyzing its behavior and summarizing (in a decision tree) the rules that are implicitely present. Declarative specifications of the software implementing the agent's behavior are obtained, and compared with the intentions of the agent's designers to help the debugging process [11].

A Progol-based algorithm for learning action theories of a single agent (mobile robot) is described by Lorenzo and Otero [19].

Alonso and Kudenko [1,18] investigate the application of ILP and EBL to complex multi-agent domains such as conflict simulations. The learning is based on reinforcement and recomputation of hypotheses, once they become out-of-date.

5.5 The York Multi-agent Environment

Some of the ideas discussed so far can be studied on a system that is being developed at the University of York [14]. The system is intended to be a general purpose Java-based platform for the simulation of learning and natural selection in a community of agents. The system draws a parallel with the world of nature. It provides tools for the specification of a two-dimensional environment with a range of features, such as different types of terrain, food and water resources. The user can edit to a large extent the specification of each of a number of *species* of agents—what they eat and fear, what types of sensors they possess, in which terrains they can hide and walk.

Single Agent Learning Each of the agents can be specified to have a Progol “mind”, *i.e.*, it can communicate with a dedicated Progol process (see figure 1). All observations collected by the agent's sensors are sent to that process as (ground) logic clauses; Progol on its turn can send back to the agent directions about its behavior. This provides a framework for single agent learning in a multi-agent environment. Learning is not limited to ILP, as the implementation of Progol includes a full-scale Prolog interpreter, which can be used for the implementation of various learning techniques. Also, both learning and recall can be carried out in an integrated way, as the newly learned clauses can be added to the Progol database. A new, multi-agent Progol is being developed at present, which, when available, will allow the agents to query each other's databases [25].

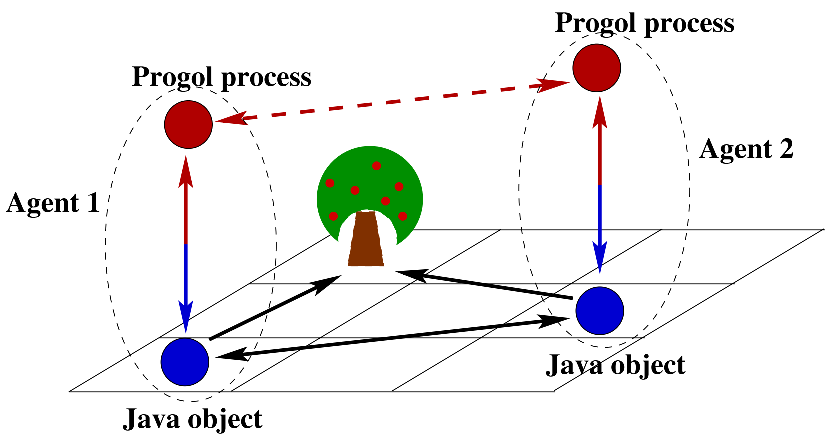
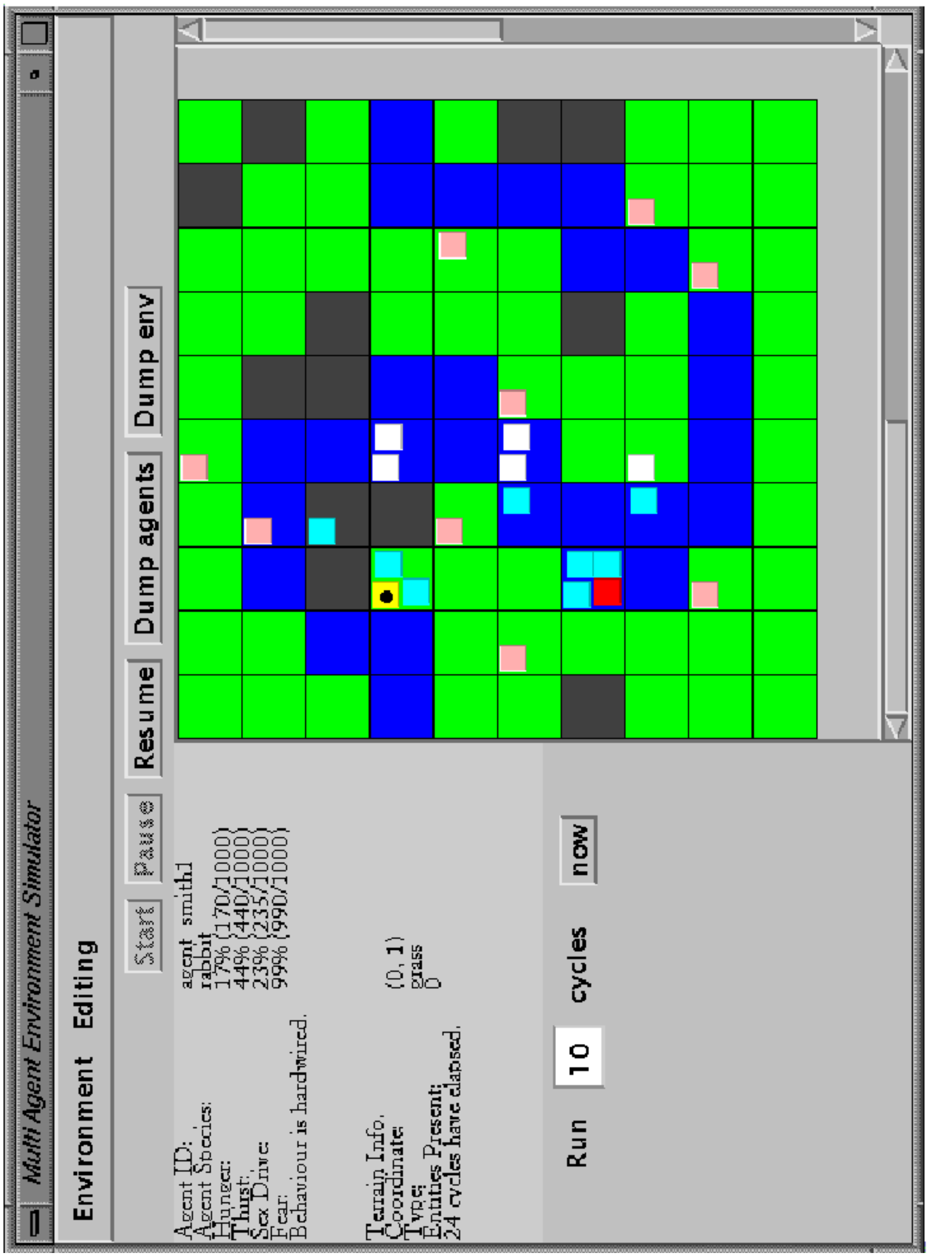


Fig. 1. Agents with a Java body and Progol mind

Synchronisation and Time Constraints To simplify the debugging and allow for easier analysis of the agents' behavior, the coordination between agents is implemented in the "One-step-at-a-time simultaneous update of environment" fashion, rather than having agents implemented as separate threads. Also, the system can easily be modified to wait indefinitely until Progol responds with an answer (selects the next action) or to allocate each agent a limited amount of time to make its move or be ignored until the next round.

Default Behavior The agents have a default, hard-coded behavior, which can be used as a baseline reference in the evaluation of learning agents or as a fallback plan if the adaptive behavior modified by learning cannot decide on an action within the required time limit. The behavior is based on the notion of "drives", which represent the intensity of each of the agent's basic needs. There are four such drives in the current implementation: hunger, thirst, fear, and sex drive. At each step, the drives are evaluated, and an action is taken to reduce the one with the highest intensity. A snapshot of an ongoing experiment is shown in figure 2. The drives of the agent marked with a black dot are shown in the left-hand side of the screen. In this case, it is a herbivor surrounded by two predators. As a result, the prevailing drive is fear, and the next action of the agent will be chosen accordingly—the agent will attempt to run away.

Natural Selection It has been mentioned that the agents have a "sex drive". Indeed, the definition of each agent includes an array of integers, which can be used in combination with the built-in genetic operators crossover and mutation to implement some of the agents' parameters as inherited features. Since reproduction involves a fixed cost, namely a contribution to the initial energy level of



```
Agent ID:        agent smith1
Agent Species:   rabbit
Hunger:          17% (170/1000)
Thirst:          44% (440/1000)
Sex Drive:       23% (235/1000)
Fear:            99% (990/1000)
Behaviour is hardwired.

Terrain Info.
Coordinate:      (0, 1)
Type:            grass
Entities Present: 0
24 cycles have elapsed.
```

Fig. 2. A snapshot of the York multi-agent environment

the offspring, natural selection will put evolutionary pressure to select the sets of parameters which improve the agent's performance.

6 Conclusion and Outlook

We presented an overview of important issues in the application of machine learning algorithms to multi-agent systems (and vice versa), starting with a description of disembodies ML algorithms, moving to single agent learning, and finally multi-agent learning (MAL). Furthermore, we discussed the application of ILP, a logic-based learning technique to MAS.

The area of MAL is very young and there is still plenty to investigate. Here are some examples of interesting future work:

Formal models of MAL: To date most developers are not able to predict the behavior of learning agents and depend purely on observing emergent patterns. Formal models of MAL that can be used to predict (or at least constrain) the behavior of learning agents would be very useful. For a first approach see [10].

More complex applications: Most MAL application domains are relatively simple. It would be interesting to see MAL research for more complex, real-world applications. Eventually, such applications would encourage researchers to look beyond Q learning.

We hope that this paper will generate interest in multi-agent learning and encourage new researchers to look into the open issues.

References

1. E. Alonso and D. Kudenko. Machine learning techniques for adaptive logic-based multi-agent systems. In *UKMAS '99*, Bristol, UK, 1999.
2. T. Balch. Behavioral diversity as multiagent cooperation. In *SPIE'99 Workshop on Multiagent Systems*, 1999.
3. T. Balch. Hierarchic social entropy: an information theoretic measure of robot team diversity. *Autonomous Robots*, July 2000.
4. C. H. Bryant and S. H. Muggleton. Closed loop machine learning. Technical Report YCS 330, University of York, Department of Computer Science, Heslington, York, YO10 5DD, UK., 2000.
5. W. Daelemans, A. V. D. Bosch, and J. Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11, 1999.
6. T. Dietterich and N. Flann. Explanation-based learning and reinforcement learning. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 176–184, 1995.
7. S. Džeroski, L. De Raedt, and H. Blockeel. Relational reinforcement learning. In D. Page, editor, *The Eighth International Conference ILP-98*, pages 11–22, Madison, Wisconsin, USA, 1998. Springer-Verlag.
8. K. Furukawa, editor. *The First International Conference on Discovery Science*, LNCS, Fukuoka, Japan, 1998. Springer-Verlag.

9. G. Gilbert and R. Conte (Eds.). *Artificial Societies: The Computer Simulation of Social Life*. UCL Press, London, 1995.
10. D. F. Gordon. Asimovian adaptive agents. *Journal of Artificial Intelligence Research*, 13:95–153, 2000.
11. N. Jacobs, K. Driessens, and L. De Raedt. Using ilp systems for verification and validation of multi-agent systems. In D. Page, editor, *The Eighth International Conference ILP-98*, pages 11–22, Madison, Wisconsin, USA, 1998. Springer-Verlag.
12. L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
13. D. Kazakov. *Natural Language Applications of Machine Learning*. PhD thesis, Czech Technical University, Prague, Czech Republic, January 2000.
14. D. Kazakov, L. Mallabone, and S. Routledge. Implementing natural selection and symbolic learning in a multi-agent environment. Manuscript.
15. D. Kazakov and S. Manandhar. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 2001. Forthcomming.
16. H. Kitano, Y. Kuniyoshi, I. Noda, M. Asada, H. Matsubara, and E. Osawa. Robocup: A challenge problem for AI. *AI Magazine*, 18:73–85, 1997.
17. M. Klusch (Ed.). *Intelligent Information Agents*. Springer-Verlag, 1999.
18. D. Kudenko and E. Alonso. Logic-based multi-agent systems for conflict simulations. In *UKMAS '00*, Oxford, UK, 2000.
19. D. Lorenzo and R. P. Otero. Using an ilp algorithm to learn logic programs for reasoning about actions. In J. Cussens and A. Frisch, editors, *Work-in-Progress Reports of ILP-2000*, pages 163–179, London, UK, 2000.
20. T. Matsui, N. Inuzuka, and H. Seki. A proposal for inductive learning agent using first-order logic. In J. Cussens and A. Frisch, editors, *Work-in-Progress Reports of ILP-2000*, pages 180–193, London, UK, 2000.
21. R. S. Michalski, I. Bratko, and M. Kubat, editors. *Machine Learning and Data Mining - Methods and Applications*. Wiley, 1998.
22. T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:4–80, 1986.
23. T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
24. R. J. Mooney and M. E. Califf. Induction of first–order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, June 1995.
25. S. Muggleton. Personal communication.
26. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
27. S. Muggleton. Learning from positive data. *Machine Learning*, 1998.
28. S. Muggleton and M. Bain. Analogical prediction. In *Proc. of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 234–244, Berlin, 1999. Springer-Verlag.
29. S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference of Machine Learning*, pages 339–352. Morgan Kaufmann, San Mateo, CA, 1988.
30. S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, Tokyo, 1990. Ohmsha.
31. S. Natarajan. *Imprecise and Approximate Computations*. Kluwer, 1995.
32. L. Parker. *Heterogeneous Multi-Robot Cooperation*. PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1994.

33. G. Plotkin. A note of inductive generalization. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 5*, pages 153–163. Edinburgh University Press, 1970.
34. F. Provost and D. Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI '96)*, 1996.
35. J. Quinlan. Learning logical definitions from relations. *ML*, 5:239–266, 1990.
36. M. Reid and M. Ryan. Using ilp to improve planning in hierarchical reinforcement learning. In J. Cussens and A. Frisch, editors, *The Tenth International Conference ILP-2000*, pages 174–190, London, UK, 2000. Springer-Verlag.
37. S. Sen and G. Weiss. Learning in multi-agent systems. In G. Weiss, editor, *Multi-Agent Systems: A modern Approach to Distributed AI*, pages 259–298. MIT Press, Cambridge, MA, 1999.
38. L. Steels. Cooperation between distributed agents through self-organization. In Y. Demazeau and J.-P. Mueller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 175–196. Elsevier Science, Amsterdam, 1990.
39. G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.
40. C. A. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In *Proc. of the Sixteenth International Machine Learning Conference*, Bled, Slovenia, 1999.
41. URL: http://verbmobil.dfki.de/.
42. J. Vidal and E. Durfee. Agents learning about agents: A framework and analysis. In *Working Notes of the AAAI-97 workshop on Multiagent Learning*, pages 71–76, 1997.
43. C. Watkins and P. Dayan. Q learning. *Machine Learning*, 8:279–292, 1992.
44. G. Weiss. Learning to coordinate actions in multiagent systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI 93)*, pages 311–316, 1993.
45. G. Weiss and P. Dillenbourg. What is 'multi' in multiagent learning? In P. Dillenbourg, editor, *Collaborative Learning. Cognitive and Computational Approaches*, pages 64–80. Pergamon Press, 1999.

Relational Reinforcement Learning

Kurt Driessens

Department of Computerscience, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Leuven, Belgium
kurt.driessens@cs.kuleuven.ac.be

Abstract. This paper presents an introduction to reinforcement learning and relational reinforcement learning at a level to be understood by students and researchers with different backgrounds.
It gives an overview of the fundamental principles and techniques of reinforcement learning without involving a rigorous deduction of the mathematics involved through the use of an example application.
Then, relational reinforcement learning is presented as a combination of reinforcement learning with relational learning. Its advantages — such as the possibility of using structural representations, making abstraction from specific goals pursued and exploiting the results of previous learning phases — are discussed.

1 Introduction

Many tasks that could be handled by today's computers aren't because there is no appropriate software available. Processes like driving a car, playing the GO board-game or more industrial applications such as controlling a manufacturing plant are not automated, not because computers don't have the computing power to handle them, but because it is too hard for software engineers to formalise in a computer program how to do it. Artificial intelligence researchers try to solve this kind of problems by making the computer learn how to handle them.

Two ways of solving such a learning task can be distinguished. Using *supervised learning* one supplies the learning system with a combination of labelled correct or optimal and wrong or non-optimal examples of state–action combinations. The learning system then tries to generalise over these examples, trying to extract the characteristics of actions or states that indicate correctness. If the system generalises correctly, it can then predict the optimality of new, unseen states and actions. Because of the fact that the learning system tries to imitate the decisions it was shown in the learning examples, this technique is often referred to as *behavioural cloning*. However, it is often very difficult (or even impossible) to supply enough correct or optimal training examples to the learning system. Automating the generation of correct examples, is as hard as designing the full controller from scratch.

Reinforcement learning is by definition unsupervised. In reinforcement learning, an agent faces the problem of learning a beneficial behaviour through his own interactions with his environment. The world the agent interacts with is regarded as a set of states. The actions the agent performs influence the state transitions and by supplying the agent with feedback about the quality of his actions through rewards of some kind, the agent can learn to optimise his behaviour.

2 Reinforcement Learning

Consider the task of learning how to play Tetris. The agent — in this case the player of the game — will try to determine what the optimal action is towards receiving the maximum number of points. The information the agent has to help him make the right decision consists of the shape of the wall on the game-field, the shape and location of the falling block and possibly the shape of the next block to be dropped. This Tetris example will be used throughout the paper to illustrate different topics.

In general a reinforcement learning problem can be described as follows:

Given

- a set of possible states S.
- a set of possible actions A.
- a *to the agent unknown* transition function δ: $S \times A \rightarrow S$.
- a *to the agent unknown* real-valued reward function $r : S \times A \rightarrow R$.

Find a policy $\pi^* : S \rightarrow A$ that maximises

$$V^{\pi}(s_t) = \sum_{i=o}^{\infty} \gamma^i r_{t+i} \tag{1}$$

for all s_t, with $0 \leq \gamma < 1$.

At each point in time, the reinforcement learning agent will be in one of the states s_t of S and selects an action $a_t = \pi(s_t) \in A$ to execute according to its policy π. Executing an action a_t in a state s_t will put the agent in a new state $s_{t+1} = \delta(s_t, a_t)$. The agent also receives a reward $r_t = r(s_t, a_t)$.

The value $V^{\pi}(s_t)$ from the description above can be thought of as the *utility of a state*. It gives an indication of the reward the agent can expect when following policy π starting from the given state s_t, with respect to the reward discount factor γ.

The task of learning is then to find an optimal policy, i.e. a policy that will maximise the discounted cumulative reward (denoted by π^*). This formulation of reinforcement learning is very similar to what can be found in the literature (cf. [8,5]).

In the Tetris example the *state* of the world is determined by the location of squares on the game-field, the shape, orientation and location of the falling block and the shape of the next block to be dropped. The *actions* the agent can influence the state of the environment with are *left, right, turn, drop* and *doNothing*. The *rewards* are defined as the number of points scored by deleting one, two, three or four lines.

Given the maximum utility (denoted by V^*) of all states and given a model of the environment, the optimal strategy can be easily obtained. In each state the agent encounters, he can choose the action that will lead him to the next state with the highest utility.

$$\pi^*(s) = argmax_a[r(s,a) + \gamma V^*(\delta(s,a))] \tag{2}$$

In this expression $argmax_a$ is a function that returns the value of a that maximises the given argument.

Several algorithms for finding the maximum utility of states exist, most of them try to compute the utility of states incrementally. For more information on these algorithms, the reader can consult [10].

2.1 Q-learning

The previous section stated that, given the maximum utility of all states and given a model of the environment, an agent could easily deduce an optimal strategy. In real applications, however the agent often does not know the transition function δ. In this case, learning the utility of states is to no avail.

Therefor, instead of learning the utility of states, an agent can learn a different function, which quantifies the quality of an action in a certain state, the *Q-function*. This function is defined as follows :

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a)) \tag{3}$$

This is exactly the value that is maximised in equation 2. Therefore, this value can be used to generate an optimal policy, this time without the need for a model of the environment.

```
for each $s$, $a$ do
    initialise each table entry $\hat{Q}(s,a)$
generate a starting state $s$
while true do
    select an action $a$ and execute it
    receive an immediate reward $r = r(s,a)$
    observe the new state $s'$
    update the table entry for $\hat{Q}(s,a)$ as follows:
        $\hat{Q}(s,a) \leftarrow r + \gamma max_{a'}\hat{Q}(s',a')$
endwhile
```

Fig. 1. The basic Q-learning algorithm.

2.2 Update Rules

An algorithm for calculating the Q-values for a deterministic Markov process [8] is given in figure 1. Several other update-rules exist, the most important ones being the adaptation of the rule for stochastic Markov processes

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha[r + \gamma max_{a'}\hat{Q}(s',a')] \; with \; \alpha = \frac{1}{1+visits(s,a)} \tag{4}$$

and the update rule used in temporal difference learning $TD(\lambda)$. [10]

$$Q^{\lambda}(s,a) \leftarrow r + \gamma[(1-\lambda)max_{a'}\hat{Q}(s',a') + \lambda Q^{\lambda}(s,a)] \tag{5}$$

One improvement that is often used to lower the time till convergence is to store the state-action pairs together with their appropriate rewards until the agent reaches a goal-state, and then learn on the encountered states and actions backwards. This allows the Q-values to spread more rapidly throughout the state space. One series of state-action pairs is called an *episode.*

2.3 Exploration versus Exploitation

Q-learning is exploration insensitive. This means that the Q-values converge to the correct values no matter what policy is used to select actions, provided that each state-action pair is visited often enough.

Several mechanisms for selecting an action during the execution of the Q-learning algorithm exist. For a discussion of a few formally justified techniques the reader can consult [5]. Usually, more ad hoc techniques are used. Greedy strategies choose the action with the highest Q-value to optimise the expected pay-back. However, to ensure a sufficient amount of exploration to satisfy the convergence constraints of the Q-learning algorithm often one has to degrade to an ϵ-greedy approach, where instead of the optimal action, a random action is chosen with a probability ϵ.

Randomised selection techniques include for example *Boltzmann exploration* where an action is chosen according to the following probability distribution:

$$P(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{a'} e^{Q(s,a')/T}} \tag{6}$$

The *temperature* T can be lowered over time. A low temperature places a high emphasis on the calculated Q-values.

Interval based exploration calculates upper bounds for a confidence interval for the Q-values for each action in a state and then chooses the action with the highest upper bound. This ensures initial exploration of states but decreases the exploration when confidence in the calculated Q-values grows.

2.4 Generalisations

The previous discussion assumed the use of some form of lookup table to represent the Q-function. This limits the use of Q-learning, not only by requiring impractical amounts of memory space, but also because it does not allow the learner to generalise over his experiences so far. In the Tetris example, one can expect the experience of scoring a tetris by dropping a block on column 7, to benefit the player when he has a similar opportunity at column 4.

Several generalisation techniques exist. Neural networks such as used by Tesauro in his famous TD-Gammon program for playing backgammon [11], are probably the most popular in reinforcement learning today. In the next section we discuss a technique that uses first order regression trees.

3 Relational Reinforcement Learning

Relational reinforcement learning is a learning technique that combines reinforcement learning with relational learning or inductive logic programming. Due to the use of a more expressive representation language to represent states, actions and Q-functions, relational reinforcement learning can be potentially applied to a wider range of learning tasks than conventional reinforcement learning. In particular, relational reinforcement learning allows the use of structural representations, the abstraction from specific goals and the exploitation of results from previous learning phases when addressing new (more complex) situations.

3.1 Relational Representation

The table-representation discussed in the previous session is impractical for all but the smallest state-spaces. The Q-table for the Tetris example would require approximately 10^{60} different values, a great deal of which will almost never be reached and with large parts of the table dealing with very similar state-action situations. Furthermore, using look-up tables does not work for infinite state spaces which could arise when first order representations are used.

Despite the fact that the use of a relational representation for states and actions is possible when using a lookup table, this method is unable to capture the structural aspects of the planning task and use them as a starting point for possible generalisations over encountered states and actions. In the Tetris game, a very well known feature of a game-state is the existence, location, depth and width of a canyon. (See figure 2) Being able to grasp this kind of features of a game-state can greatly reduce the number of different states that need to be represented.

Whenever the goal is changed a table-representation would require retraining the whole Q-function. In the Tetris game for example, one hopes to be able to use the training experience for placing an L-shaped block on column 2 when placing a T-shaped block on column 5.

Also, one would expect that experience gained in a small state-space — which is usually easier and faster to learn in — could be (partly) recycled when learning on a larger, more complicated state-space. It is unclear how to achieve this with a lookup table.

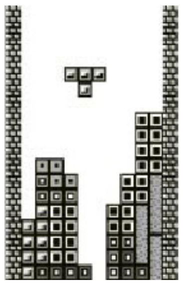


Fig. 2. A typical Tetris game-field with two “canyons”.

The need for a lookup-table can be resolved by using an inductive learning algorithm (e.g., a neural network as in [11]) to approximate the Q-function. The other problems — capturing structural aspects, goal-abstaction and state-space abstraction — can only be solved by using a *relational* learning algorithm that can abstract from specific states and actions using variables. Such a relational learning algorithm called RRL is now presented.

3.2 RRL-algorithm

In the RRL-system [4] the Q-function is only updated at the end of each episode, as described in section 2.2. Instead of updating the table entries for the encountered state-action pairs, the entire episode is offered to a learning algorithm called TG, which incorporates the new examples in its representation of the Q-function.

The TG-algorithm represents the Q-function as a *logical regression tree*. A regression tree is a decision tree in which the leaves of the tree contain a continuous (real) class value. To make predictions with a decision tree one starts in the root of the tree and applies the root’s test to the example. Next, the branch that corresponds to the outcome of the test in the example is taken and the example propagates to the corresponding subtree. If the resulting subtree happens to be a leaf, one reads off the prediction. See figure 3 for an example of a logical decision tree that indicates whether there is a possibility to score a tetris. (It will not find all tetris scoring opportunities however.)

Classical decision trees employ propositional or attribute value representations. Recently, however, these representations have been upgraded towards first order logic, resulting in the frameworks of logical classification and regression trees [6,2]. One key difference between logical and classical decision trees is that classical decision trees work with examples in attribute value form and logical decision trees can use examples which are a relational database (or a Prolog knowledge base) described by a set of facts. Logical decision trees can introduce variables in the tests that they use and reference these variables in tests at lower nodes. RRL uses logical decision trees as implemented in the programs TILDE [2] (for classification) and TILDE-RT [1] (for regression).

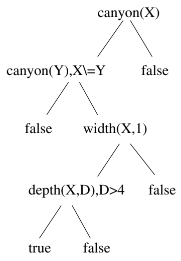


Fig. 3. A Logical Decision Tree

3.3 TG-algorithm

Classification and regression trees are typically induced using a divide and conquer algorithm, called top-down induction of decision trees (TDIDT). The reader can consult [9] or [2] for more information. However, since reinforcement learning is an incremental process, this kind of algorithm does not suffice for a reinforcement learning system. The RRL-system uses an upgraded version of the G-tree-algorithm [3] called TG, which differs from the G-tree-algorithm only because it is able to handle first order representations. Figure 4 presents a high-level view of the TG-algorithm. A leaf stores for each candidate test the number of examples for which the test succeeds, the sum of their Q-values and the sum of their squared Q-values. The same three numbers are stored for the set of examples for which the test fails. These six numbers enable the TG-algorithm to compute the significance of a test in a leaf and decide whether that leaf needs to be split. A more rigorous description of the G-tree-algorithm can be found in [3].

```
create an empty leaf
while data available do
    split data down to leafs
    update data in leaf
    if split needed then
        grow two empty leafs
endwhile
```

Fig. 4. The TG-algorithm.

3.4 The P-RRL Algorithm

The previous sections illustrated how RRL constructs a Q-tree to represent the Q-function. From this Q-tree, a policy can then be derived. One way of representing this policy is through the use of a P-function, i.e. a function which returns true or a 1 if a given action is optimal in a given state, and false or 0 otherwise.

Due to the representation power of first order logic, it is possible to represent this P-function as a logical decision tree as well. As the Q-function is typically more complex than the P-function — a Q-function implicitly codes the distance to and the amount of the next and later rewards — the learning of the P-tree usually leads to a further improvement of the generated policy or faster convergence to the optimal policy.

The P-tree is learned starting from the Q-tree. For each state visited in an episode, RRL looks at all the actions that are possible in this state and the Q-value that the Q-tree predicts for these actions. From these values examples are generated of optimal and non-optimal state–action examples which are then in turn used to generate the P-tree with the TG algorithm discussed before.

4 Experiments

To demonstrate the behaviour of RRL, a few tests will be discussed here. The blocks world [7] is used as a test-case, because it is easy to understand and it exhibits some characteristics that demonstrate some specific RRL advantages.

Three goals were studied:

1. stacking all blocks
2. unstacking all blocks
3. stacking two specific blocks: on(X,Y). Keep in mind that X and Y are variables and can be substituted for any two blocks after learning.

Table 1. Number of states and number of reachable goal states for three goals and different numbers of blocks.

No. of blocks	No. of states	RGS stack	RGS on(a,b)	RGS unstack
3	13	6	2	1
4	73	24	7	1
5	501	120	34	1
6	4 051	720	209	1
10	58 941 091	3 628 800	1 441 729	1

As one can see in table 1 the blocks world becomes intractable very rapidly when the number of blocks increases. Because the goal of the experiments was to learn policies which would work for any number of blocks each episode RRL was supplied with a different amount of blocks to train on. Because 6 blocks already imply a large state space for which the chances of reaching the goal state decrease rapidly (especially for the unstacking goal), the amount of blocks was varied from 3 to 5. P-learning was started after 2500 episodes. The strategies generated by RRL were tested on 5000 test-cases in which the amount of blocks varied from 3 to 10.

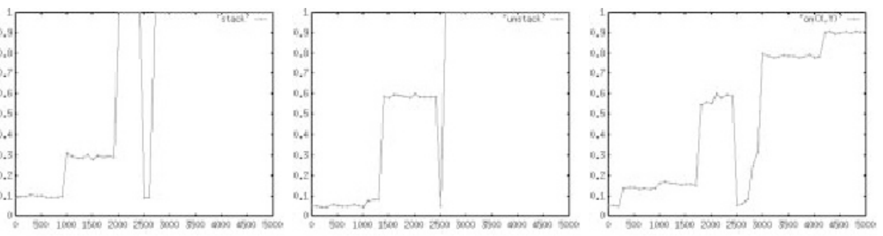


Fig. 5. The learning curves for the blocks world. P-learning starts after 2500 episodes.

Figure 5 shows the learning curves for the three experiments. The curve for stacking shows that RRL is able to generalise the Q-function for stacking 3 to 5 blocks to work for worlds with up to 10 blocks. After a short dip when starting to learn and test the P-tree, also the P-function generates an optimal policy for states up to 10 blocks. The learning curve for unstacking does not reach optimality before P-learning is started, but it does so very rapidly afterwards. The learning curve for stacking two specific blocks does not reach optimality, not even after P-learning starts. For this goal, it is very hard to represent the optimal policy as a decision tree.

5 Concluding Remarks

This paper gave a brief introduction to reinforcement learning and relational reinforcement learning. The advantages of RRL include the ability to use structured representations, which enables learning in very large or even infinite worlds, and the ability to use variables, which allows the system to abstract away from specific details of situations.

The current use of logical decision trees is not required for the applicability of RRL. Future work will certainly include the exploration of other first order representations of the Q-function and possibly the P-function. There will also be a further exploration of the application domains for RRL. One example application that is being looked at is graph colouring.

Acknowledgements

The author would like to thank Sašo Džeroski and Luc De Raedt who started the work on Relational Reinforcement Learning and Jan Ramon for his help with the implementation of the TG-algorithm.

References

1. Blockeel, H., De Raedt, L., and Ramon, J. Top-down Induction of Clustering Trees. In *Proc. 15th Intl. Conf. on Machine Learning*, 1998.
2. Blockeel, H., and De Raedt, L. Top-down induction of first order logical decision trees, *Artificial Intelligence*, 101(1-2):285-297, 1998.
3. Chapman, D., and Kaelbling, L. Input generalisation in delayed reinforcement learning: An algorithm and performance comparisons. In *Proc. 12th Intl. Joint Conf. on Artificial Intelligence*, 1991, Morgan Kaufmann, San Mateo, CA.
4. Džeroski, S., De Raedt, L., and Driessens, K. Relational Reinforcement Learning, Machine Learning, Vol. 43, 2001, in press.
5. Kaelbling, L., Littman, M., and Moore, A. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4: 237–285, 1996.
6. Kramer, S. Structural regression trees. In *Proc. 13th Natl. Conf. on Artificial Intelligence.* AAAI Press, 1996, Menlo Park, CA.
7. Langley, P. *Elements of Machine Learning.* 1996, Morgan Kaufmann.
8. Mitchell, T. *Machine Learning.* 1997, McGraw-Hill, New York.
9. Quinlan, J.R. Induction of decision trees. *Machine Learning*, 1: 81-106, 1986.
10. Sutton, R. and Barto A. *Reinforcement Learning: an introduction.* 1998, MIT Press.
11. Tesauro, G. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

From Statistics to Emergence: Exercises in Systems Modularity[1]

Jozef Kelemen

Institute of Computer Science, Silesian University
746 01 Opava, Czech Republic
e-mail: kelemen@fpf.slu.cz

Abstract: The contribution sketches several ways of considering systems from the position of their modularity through viewing systems without any attention focused to their modularization, then as composed from functionally specified modules, up to the post-modular systems consisting of relatively independent autonomous modules sharing a common environment and acting in it. A relatively simple, uniform and productive theoretical framework for study of the mentioned aspects of systems behavior and modularity – the framework of the theory of grammar systems – will be presented, illustrated and discussed in certain details.

1 Introduction

Considering complex *systems* we typically need – cf. [40] – to know *what* a complex system is doing at a higher level in order to find out *how* at the lower level it accomplishes that task. In other words, we often need to know the function of the complex system being analyzed to know what aspects of structure, of system architecture, to look at. Understanding the behavior of a complex system requires knowing which aspects of the complex mass of lower-level properties are significant in making a contribution to the overall behavior of the system. In real situation such distinction may be problematic because the identification of which aspects of the lower-level activity of the system are significant and which are noise – the process of the so-called distillation [40] – is often complicated.

In following sections we will sketch several ways of considering complex systems from the position of their modularity through viewing systems without any attention paid to their modularization, then as composed from functionally specified modules, up to the post-modular systems consisting of relatively independent autonomous modules sharing a common environment and acting in more or less coordinated ways.

Especially in the case of post-modular systems becomes the emergence of their behavior from massively parallel interactions of modules with the environment critical. We discuss this topic in more details in Sec. 7 and relate a proposed understanding of emergence to the test of emergence proposed in [46]. Sec. 8

[1] Research supported by the Grant Agency of the Czech Republic, grant No. 201/99/1086.

provides an example how simple rational behavior of a post-modular system emerges from behaviors of individually non-rational modules interacting through a shared environment, only.

Another difficult problem is to find the right boundaries of a complex system; cf. [3]. The solution is crucial for identification of the loci of control of the complex systems. There are two extremes here:
To consider the environment of the system as the locus of control -- the case of *external control*, and to consider the system itself as the locus of its own control -- the case of *internal control.*

The result of emphasizing the external factors is, according [3], to reduce the importance of the system, treating it as responding to external factors, or shaped by them, but not itself an important element in accounting for the responses. On the other hand, limited responsiveness in the face of wide environmental variation it taken, as an indicative of internal control, and the solution is to search for specialized and complex internal mechanisms. The system makes its own contribution and influences what happens to it. We will demonstrate the possibility to consider the same systems as belonging to different categories according this taxonomy.

A relatively simple, uniform and productive theoretical framework for study of the mentioned aspects of systems behavior and modularity – the framework of the theory of *grammar systems* – will be presented, illustrated and discussed in certain details, and necessary bibliographical information will be provided for those having deeper interest in the presented subject and approach to dealing with it.

The presentation will be focused to six main topics:

1. An example of a simple behavior of a system will be presented in Sec. 2.
2. Use of statistics in order to construct models of systems on the base of observations of their behaviors (in the form of Markov-processes and/or probabilistic finite-state machines as models of systems) will be sketched in Sec. 3.
3. Functional modularization: looking into the black box and construction of functionally specified modules and their addressed interrelations in order to generate the (almost) required behavior and proposing a traditional language-theoretic approache to model behaviors by generative devices will be the subject of Sec. 4.
4. Post-modularity: considering systems as societies of agents and their functional modularization into the form of so called post-modular systems will be presented in Sec. 5; Sec. 6 will provide a (very) short introduction to the theory of grammar systems as a formal model of post-modular systems.
5.The problem of emergence: the emergence test applied to post-modular systems and languages generated by grammar systems as an example of emergence of behaviors will be described in Sec. 7.
6. Emergence of low-level rationality: in Sec. 8, an example will be described how a specific low level rationality of behavior can emerge from the non-rational behaviors of modules in post-modular systems.

The contribution ends with a relatively exhaustive list of references which provides the basic orientation in the field of colonies – a sub-field of the theory of grammar systems – used in this article as the main conceptual framework for dealing with the mentioned problems with all theoretic rigor.

2 An Example

We will start and in the following Section in certain extent also follow an example proposed in [45]. Let us suppose that we observe the behavior of a particular system and that we realize that the system reacts to a starting stimulus (symbol) S by producing a strings of symbols a and b of the form $a^n b^n$ (for *some* finite integer n). We hypothesize from some reasons that the behavior of the system will be of the same structure for *all* positive n. However, we know nothing about the architecture of the system, so the system is something like a black box for us. The above-described situation is visualized in Fig. 1.

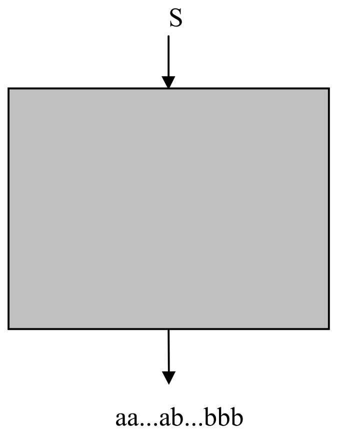


Fig. 1: A system as a black box

Let us consider now, that we are confronted with a problem to reconstruct this system on the base of its observed behavior. How we can proceed?

Next sections offer some answers to the previous question. We show how hard is to start from a finite sample of observations and invent in such a base a device with an expected infinite behavior. Realizing that we will see that the requirement of an infinite behavior can lead to different system architecture and that it is hard to decide what system constructed on the base of the same finite sample of behavior is the most suitable one.

3 A Simple Statistical Analysis

Consider the system form the previous example. The sample of observed behavior can be considered as a corpus of a very simple language. Le us assume that this sample – consisting in, say, 341 observations – yield the record in Table 1.

ab	200x
aabb	100x
aaabbb	30x
aaaabbbb	10x
aaaaabbbbb	1x

Table 1: The summary of observation of behavior of the system from Fig. 1

Let us to concentrate in that table to the neighboring strings appearing in the sequences of symbols in strings from Table 1. Let the dot • denotes the „head“ and the „tail“ of each string. Then the following statistics follows:

<u>The Statistics</u>

neighborhood:	*times:*
(•a)	341
(a•)	0
(ab)	341
(•b)	0
(b•)	341
(aa)	194
(bb)	194
(ba)	0

If we wish to reconstruct the behaving system on the base of this record, so on the base of a finite corpus. One way might be to describe the strings as segments of a Markov chain with four states (a, b, and the boundary states •). Such a description would be summarized as a transition matrix giving the probability of a transition between any pairs of symbols as it is in the Table 2:

	•	a	b
•	0.00	1.00	0.00
a	0.00	0.36	0.64
b	0.64	0.00	0.36

Table 2: The table of probabilities of neighborhoods.

Fig. 2 represents schematically an other type of finite state device – a finite automaton – which generates the sequences of symbols a and b with similar statistical distribution as the above described Markov chain

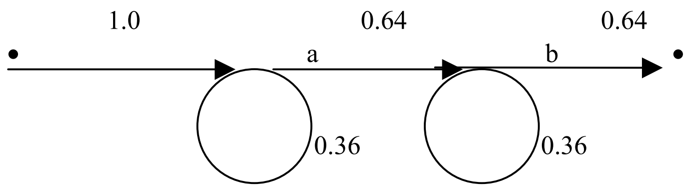


Fig. 2: A finite state device generating sequences of *a*s and *b*s.

The proposed models give a very simple characterization of the corpus but suffer from certain weakness. For example, while the first device predicts that ab will be the most common string, it also predicts that aab and abb should occur with greater frequency than aabb. As Pylyshyn [45] discussed in more details, this can be improved e.g. by making the states of the Markov process correspond to pairs of symbols, or to even larger sequences. Such model, however, begin lose its compactness and depends on more and more empirical parameters. Instead of the above-described performance-based approach to construct the behaving systems, in the next Section we will follow another methodology.

4 Traditional Modularization

We can criticize the models from the previous Section because of their specific inadequacy. Imagine, for instance, that the symbols a and b denote some elementary physical processes like go one step downstairs and go one step upstairs, respectively. All the observed behavior of the system then means to go some steps downstairs and then go back the same number of steps upstairs. So, the symmetry which we can recognize on the base of the previous corpus (and the importance of which is depressed in the Markov model) may easily become as an important requirement.

An almost generally accepted traditional conviction in systems design is that systems can be modularized into some *functionally* specified units (or modules). In fact, decomposition of systems into functional modules is such basic principle that we have tend to take it granted. The traditional computer science reflects this position in the form of the so-called procedural abstraction. From this position, computational units (procedures) are defined as functions, which generate from given inputs the corresponding outputs. In Fig. 3 the basic idea of internal structure of the required system is depicted.

$$\begin{tikzcd}
S \arrow[d] \\
\textbf{i t e r a t i o n} \arrow[d] \\
\textbf{t e r m i n a t i o n} \arrow[d] \\
\text{aaa...ab...bbb}
\end{tikzcd}$$

Fig. 3: A hypothesized functional modularization of the system from Fig. 1

To generate the required set of strings we can imagine a module presented by the rule $S \to aSb$, which makes the activity of another module $S \to ab$ possible, and which terminates the generative process. These two modules/rules will work sequentially during the rewriting process in order to generate a string of the required structural properties (see Fig. 4).

$$\begin{tikzcd}
S \arrow[d] \\
S \to aSb \arrow[d] \arrow[loop left] \\
\text{a...aSb...b} \arrow[d] \\
S \to ab \arrow[d] \\
\text{aaa...ab...bbb}
\end{tikzcd}$$

Fig. 4: A functional modularization of the system from Fig. 1

This style of thinking and the resulting system decomposition is usually called as the system decomposition into *functionally* specified modules. As Fig. 4 illustrates, e.g. all the traditional formal grammars studied in the literature on theoretical computer science are in certain sense examples of functionally modularized generative systems, which generate set of strings of symbols – the formal languages. But while functional decomposition is certainly a legitimate architecture principle, it is not the only one applied in the present time.

5 Why to Proceed in an Other Way?

The traditional view that systems are set up from *functionally specifiable modules* which interacts in some well-specified ways is rooted in our traditional approach to systems (re) construction consisting in the decomposition (analysis, modularization) of the whole (required) system into a set of functionally specified modules, in constructing such modules and in establishing their well-specified interrelatedness which guaranties the required functioning of the whole system set up in such a way.

However, the above-described methodology has some limitations, and is effectively applicable for constructing *not too complex* systems, only. Too complex systems are those, at least in the present context, which have, for instance too many modules, and we do not know their functional specification. However, we are interested in keeping some functional specification of such systems in certain limiting boundaries. Eco-systems are perhaps a good example. Or imagine systems, which functional decomposition is unknown for us, and behavior of which cannot be changed in another way than by deleting and/or adding some new components. Different societies are good enough examples. We do not know exactly what a particular member of a society will do in a specific situation, but we are interested in acceptable functioning of the whole society in the same situation.

In [54] several good reasons are listed to suppose that many systems are neither analyzable nor synthetizable from the positions of functional decomposition scheme. This follows, in general, mainly from the *interactive* nature of some non-computational systems (like behavior-based robots or economic systems) but increasingly also from many computational systems (software agents or computer networks, for instance). The common characteristics of such systems are their massive mutual interaction and/or massive interaction with dynamically changing environments in which there are (physically or informationaly) situated. Fig. 5 presents a view of a post modular architecture of the system depicted in Fig. 1.

Realize that the two components of this system communicate through the shared environment only, and that there is no direct communication between them. However, if the case of an appropriate „starting“ situation in the environment, esp. in the situation when only one symbol A appears in it, the behavior (sketched in the Fig. 6) of the whole system is exactly the same, as required in the Fig. 1 and as is produced by the traditionally designed system depicted in Fig. 4.

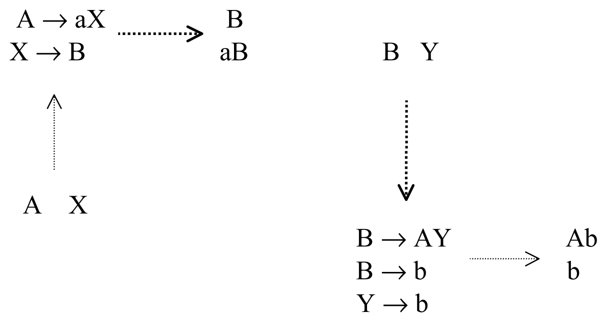


Fig. 5: A post-modular architecture of the system from Fig. 1

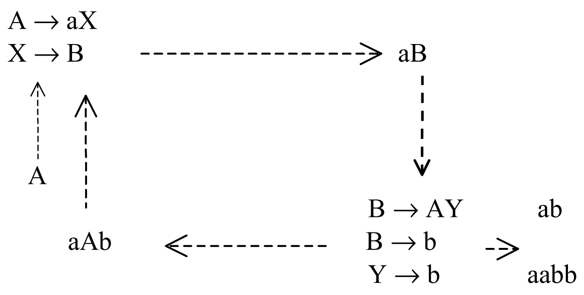


Fig. 6: Interactions of modules in a post-modular system

Generally, an *interactive system* is – according Stein – one in which information is continually being sensed and behavior is continually being generated. Interactive systems exploit properties of their environment to generate future behaviors. They are always operating concurrently with their environment. While traditional computer procedures – while executing – are blissfully ignorant of their environment, writes Stein [54], an interactive system, in contrast, is always responding to its environment. It means also, that interactive systems are opportunistic in the sense that they depend on simpler systems without necessarily preserving modular-functional encapsulation with all of the redundancies, interdependence, etc. Fortunately, this does not mean that we must give up all our conviction in principled analysis or synthesis. The basic principle derived from experiences is, according [54], that the post-modular systems are set up opportunistically from parts working in the massive *interaction* with their environment(s) where interactive systems exploit their environments to generate future behaviors, and that the behavior of the whole post-modular system *emerges* from this interaction.

6 A Formal Framework

Rather than imagine and develop completely new conceptual approaches to understand post-modular systems theoretically, we propose to reuse at least some of existing ones. There are many reasons to proceed in such a way. We recognize the existence of working experiences, the educated researchers, and the necessary interrelatedness with the traditional approaches as some among the most important ones. The aim of this Section is to show the possibility of effective dealing with post-modular systems using the formal framework of the theory of grammar systems satisfying the previous requirements.

Grammar systems [8], [11] define languages through finite sets of formal grammars working in shared strings of symbols (similarly as in the example depicted in Fig. 4).

Development of grammar systems has followed in the past the crucial conceptual changes in viewing computing systems. An amount of inspiring work – with results presented in more than two hundreds of papers or published conference contributions, up to three hundreds of references of the book [8] – has been done during a decade of activities in the field documented e.g. in [42], [31], [11], [33], [25], [43], [16], [28]. We suppose the theory of grammar systems as a promising way of study post-modular systems in the general framework of the theoretical computer science.

The simplest architecture realizing the idea of total decentralization expressed in grammar systems is the architecture of the so-called *colonies*. There is perhaps the simplest variant of grammar systems and it is a reason why we will deal with them in this Section. However, all the field of grammar systems can be, according our opinion, used for better understanding of some of the aspects of post-modular systems.

Originally, colonies have been invented in [27] as formal generative models of purely reactive agents studied in robotics [7], [6], [1] and have been developed during the nineties in several directions e.g. in [10], [35], [10], [42], [4], [37], [38], [34], [18], [19], [29], [30] etc. In [5] colonies are connected with a modification of the Turing machine, Sosík [51] connects them with neural networks and in [52] formulates some possibilities of inductive inference of colonies. In [22] colonies are connected with the field of reactive robotics, while in [32] and [26] with rule-based systems.

Formally, a *colony* C is a triple (R, V, T) set up from:
a finite number of components R_i belonging to the set R of *components* of C,
a finite set of symbols V called the *total alphabet* of C,
and a finite *terminal alphabet* $T \subseteq V$ of the colony C.

Components $R_i \in R$ $(1 \le i \le n)$ of a colony C are *regular grammars generating finite languages* and operating on a shared string of symbols – the *environment* of the colony – without any explicitly predefined strategy of cooperation of the components. A terminal symbol of one component can occur as a non-terminal symbol of another one.

An *environment* of a colony is formed by strings of symbols from a finite alphabet V (the environment is, in fact, the structure identical with the so called *sentential form* known in the classical theory of formal grammars and languages). Let suppose that strings are modified only by *sequential* activities of components of a colony. (However, in [10] the parallel case is also invented and studied.) Because of the lack of any predefined strategy of cooperation of components, each component participates in the rewriting of the current string, whenever its start-symbol is present in the actual string. Conflicts are solved non-deterministically, as it is usual in the classical theory of formal grammars.

The *language* L_C generated by a colony C is formed as a subset of all strings over the terminal alphabet T, which are generated by the components from R of the colony C from a given starting string. A terminal symbol of one component can occur as a non-terminal symbol of another one, so the possible cooperation of components of the colony allows generating by colonies substantially more than finite languages.

As an example of a colony C, let us consider:
A two elements set R of components R_1 working over its total alphabet $\{A, X, B, a\}$ and terminal alphabet $\{a, B\}$ according the rules $\{A \rightarrow aX, X \rightarrow B\}$, and starting with its starting non-terminal symbol A, and R_2 working over its total alphabet $\{A, B, Y, b\}$, with the terminal alphabet $\{A, a, b\}$, and using the set of rules $\{B \rightarrow AY, B \rightarrow b, Y \rightarrow b\}$ starting from its own starting non-terminal symbol B.
The total alphabet of the colony $V = \{A, B, X, Y, a, b\}$.
The terminal alphabet of the colony $T = \{a, b\}$.

It is easy to see that starting with the non-terminal symbols A of C, the colony C generates the language $L(C) = \{ a^n b^n \mid n \geq 1\}$.

It is interesting to realize in our context that $L(C)$ is an *infinite* language, while both of the two components of the colony C, the component grammars R_1 and R_2 generate *finite* languages $L(R_1) = \{aB\}$ and $L(R_2) = \{Ab, b\}$, only.

Grammar systems and colonies are formal devices, which express important features of post-modular systems postulated by L. A. Stein [54] – namely the *representational alignment through shared grounding*.

Realize first, that components of a grammar system or of a colony are really situated in their environments, of course in very simple way. However, if „situatedness" is understood as exploration of the environment for generating components own behavior, in the case of grammar systems or colonies components we are confronted exactly with that.

Second, making possible for the components of a grammar system (or of a colony) to interact with a shared environment – with the shared sentential form, in this case – a we can obtain (lot of proved theorems – e. g. in [8] – show this possibility in the theory of grammar systems and colonies) completely different behaviors (languages) as the behaviors (languages) of the components. In the case of colonies this fact is very spectacular, because from the interaction of components – which generate finite languages only – with a shared environment we receive infinite behaviors. Moreover, having in the mind the fact that an arbitrary finite language is a regular language in the famous *Chomsky hierarchy* of languages, colonies define on the base of such finite languages the whole family of context-free languages – a considerably larger family of languages in this hierarchy containing all the family of regular languages. Putting some additional – simple and very natural – conditions on the capabilities of components and without changing their individual generative capacities, the generative capacity of colonies can be enlarged also to some of the context-sensitive languages, cf. [27], [4], [29].

So, colonies enable us to prove formally the fact that, as Stein [54] states, we can capitalize on the observation that interactive systems situated in a shared environment often rely on some of the environment regularities, putting aspects of two systems into alignment, to bootstrap a shared „understanding“ between the two systems.

More spectacular – but technically more complicated – is the formal framework of the so-called *eco-grammar systems*. This framework has been inspired by some of the efforts of Artificial Life, and suggested in order to express some life-like properties of decentralized symbol systems [9]. Eco-grammar systems represent a modification of grammar systems in several aspects. One among the important feature of eco-grammar systems is that the environment in which the components act has also its own dynamics and that this dynamics is independent on the components activities. Components are described as formal grammars influencing the environment (represented by a string of symbols). In some variant of eco-grammar systems the direct interactions between components are allowed, too. Moreover, some internal changes (development) of components are provided in dependence on the environment states and dynamics.

7 Emergence

In this Section we will discuss the phenomenon of *emergence* widely recognized in numerous and quite different contexts [20] and the *emergence test* proposed in [46]. After some necessary notes concerning *emergence* we will try to prolong by another example the list of examples of emergent phenomena provided in [46]. We will show that *complicated formal languages* can emerge (in the sense required by the emergence test) from *interactions* of *simple generative devices*. In our case, all the notions written in italics are precisely, formally definable. The meaning of the words *emergence* and *to emerge* will be used to characterize a generative process *of a*, and its result – *the* language, resp.

In [20] *emergent* is explained as „... a product of coupled, context-dependent interactions. Technically these interactions, and the resulting system, are *nonlinear*: The behavior of the overall system *cannot* be obtained by *summing* the behaviors of its constituent parts... However, we *can* reduce the behavior of the whole to the lawful behavior of its parts, *if* we take nonlinear interactions into account.“

The example of a colony provided in the previous Section perhaps demonstrates that grammar systems are nonlinear in the above mentioned sense: The *infinite* language generated by the whole colony cannot by generated by its two component grammars in the sense of „summing up“ the two *finite* languages generated by each of them. However, we can consider the: summing up: operation not only as simple as in the set union. We can consider operations defining the so-called abstract families of languages. Then we may look for „emergence“ (generation of languages by grammar systems) which cannot by defined by these operation over the languages generated by component grammars of grammar systems; [17] will provide the first result of the such direction of our research.

In [15], the *emergent computation* is supposed consisting of: (1) a collection of agents, each following explicit *instructions*. (2) *Interactions* among the agents (according to the instructions), which form implicit global patterns at the macroscopic level i.e. epiphenomena. (3) A natural *interpretation* of the epiphenomena as computation.

It is clear, that all component grammar in a colony follows explicit *instructions*. These instructions are formulated in the form of rewriting rules and the process of their use is formulated explicitly in the definition of the derivation step. *Interactions* among the component grammars are limited by the strategy of cooperation and their behavior „at the macroscopic level“, so if we observe the behavior of the whole colony instead of the individual behaviors of its components, produces – as an epiphenomenon – an infinite language, which is naturally *interpretable* as a computation: it can be generated/accepted by the corresponding type of an abstract computing device (if it is a context-free language, which is the typical case for colonies, then by the corresponding push/down automaton).

Ronald, Sipper and Capcarrere in [46] have formulated a *test of emergence* trying to offer an operant definition of emergence for artificial life experiments. The requirements putted onto a system in which the phenomenon of emergence appears are the following:

Design. The designer designs the systems by describing *local* interactions between components in a language L_1.

Observations. The observer describes global behaviors of the running system using a language L_2.

Surprise. The language of design L_1 and the language of observation L_2 are distinct, and the causal link between the elementary interactions programmed in L_1, and the observations observed in L_2 are non-obvious.

Imagine the situation described in the example in Section 3 and analyze the *design* process in more details, first. Imagine a designer (or more designers) who „programs“ (constructively defines) the particular simple modules, say R_1 and R_2, for generating some required finite numbers of strings, say *{aB} and {Ab, b}*. The designer is satisfied because the constructed modules work fits the given requirements. Using both of these generative devices in isolation, they generate the simple sum of the two behaviors – the *finite* language *{aB, Ab, b}*. No surprise during the observation of such kind of system created from the (just described) *isolated* modules.

Now, imagine that a designer put the modules into a shared environment consisting of a symbol *A* (cf. Fig. 6). What they will *observe*? The global behavior of the whole systems will be other, then the simple finite „sum" *{aB, Ab, b}* of the behaviors of individual components. The observed behavior will be an *infinite* language $\{a^n b^n \mid n \geq 1\}$. However, the design of the two components remains unchanged! So, while the language of the *designer* has been concerned with *finite* behaviors, the language of the *observer* of the systems created in a rather simple way form the designed components will be concerned with *infinite* behaviors (languages).

The impression of a *surprise* becomes realizing the just demonstrated finite/infinite gap between the behaviors of the above mentioned two systems set up from the same components but working in different environments (starting with different starting strings). But theoretically, some more surprising facts are proved concerning colonies. As we have mentioned already, colonies – devices set up from regular grammars generating finite languages – define the whole family of context-free languages, and if they work with some simple and very „natural" additional restrictions, they generative capacity can be enlarged also to some of the context-sensitive languages. Moreover, in the case of the finite languages some problems seems to be highly actual, for instance the complexity of defining such languages, and some of them are completely trivial, like problems concerning decidability, computability, etc. Some questions become to be meaningful only in the case of infinite languages. In other words, there are two (not completely distinct, of course) languages for talking (and thinking) about finite languages and devices generating them, and the infinite ones.

Now, we could state that colonies satisfy the test of emergence as proposed in [46], and that they offer a quite simple and theoretically well grounded example of how the test can be applied. In such a case, moreover, we can demonstrate the appearance of the phenomenon of the emergence in theoretically (mathematically) well-described formal systems, too, not only in experimental situations discussed in [46].

8 Rationality

Intuitively, an agent couples perceptions with actions *rationally* if it is able to decide about what actions to perform in a given situation in light of its goals. It has become usual to distinguish between rational thought and rational action, or – according [40] – between *process* and *task*, because many among those interested in the study of rationality are skeptical about the widely view that rational thought is a necessary precursor of rational action. Simon [49], [50] makes a similar distinction between *procedural* and *substantive* rationality.

Roughly, *substantive rationality* is a matter of the fit between an agent's goals and its environment. A substantively rational agent simply does the best action in the sensed situation in order to achieve its goal. It reacts by an action to the state of its environment in order to achieve the just actual goal. So, an account of the substantive rationality characterizes what the agent does with respect to the currant state of its environment. In the case of a *procedurally rational agent* we suppose a process (a certain kind of computation enabling to the agent to "deliberate") behind the choice of what to do in order to achieve the current goal with respect the just actual state of agent's environment[2]. An account of the procedural rationality may characterize this process of action selection.

In terms of its expectations and beliefs about utilities of performable acts in actual states of its environment the agent's rationality then means that the executed acts are of the *maximal expected utility* for the agent among the actions available at some instant [12]. This is the core of the formalization of rationality in the framework of the classical *decision theory*.

The mathematical way of formalization of such a concept of rationality is as follows; cf. [44]:

Let **A** be an *agent* described by a finite set A of *alternative acts* between which it must decide, a finite set O of possible *states of the world*, a function u assigning numerical values – *utilities* – to the possible states of world, and a set of *beliefs* as probabilities $p(O/A)$ of each possible state of world on each act. The *expected utility* of an act a is then defined as $e(a)=\Sigma_{o\in O}\ u(o)p(o/a)$.

A *decision problem* for the agent **A** consists in maximizing the expected utility of its acts. An agent, which is able to solve the decision problem, is a *rational agent*.

In [21] a special type of rationality – the so-called *low-level rationality* of agents is defined and studied. The idea behind that level of rationality consists of eliminating probabilities and minimizing the number of considered states of the world.

We note, that the behavior of *low-level rational* (*llr-* for short) *agents* is often labeled as *adaptive behavior* and the llr-agents are sometime called *adaptive agents*. However, from the just sketched theoretical analysis follows that adaptiveness may be considered as the lowest level of rational behavior.

Formally, an agent **A** with a finite set $A = \{a_1, a_2,, ... a_n\}$ of acts has the property of *low-level rationality* or *adaptiveness* (or it is an *llr-agent* or an *adaptive agent*) if **A** is able to solve the decision problem under the conditions that:

- **A** is able to "recognizes" only two states of its world, so $O=\{t, f\}$,

- **A** has a binary utility function defined by $u(t)=1$ and $u(f)=0$,

[2] The importance of the (asymptotically) bounded optimality of computations performed by agent's components is advocated e.g. in [48] or [47]. In the case of purely reactive agents, however, we suppose no deliberation, no iterations executed by such agents, and, therefore, all computations of agents components running in the real time.

- the belief function of **A** for given *a* is either

$p(t/a_i)= 1$ and $p(f/a_i)= 0$ or $p(t/a_i)= 0$ and $p(f/a_i)= 1$.

Consequently, the expected utility function $e(a_i)$ equals 1 for $p(t/ a_i)=1$ and have the value 0 for $p(t/ a_i)=0$.

The *behavior* $L_{\mathbf{A}}$ of an llr-agent **A** is the set of all sequences $a_1...a_k$ ($k \geq 1$) of acts of **A** such that $e(a_1...a_k) = 1$ where $e(a_1...a_k) = e(a_1)...e(a_k)$.

Consider now a very simple example of an agent – a child-toy of the form of a fully mechanical ladybug see Fig. 7):

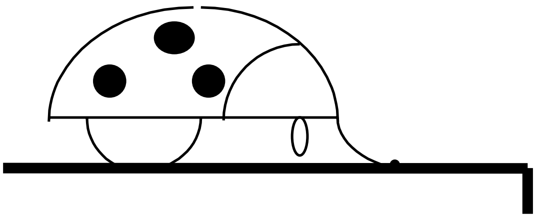


Fig. 7: The LADYBUG – a simple mechanical child-toy.

The set *A* of the possible acts of the LADYBUG contains three elements: *go*, *turn* and *stop* which will be designated in the following by *g*, *t*, and *s*, respectively. The set *S* of possible situations contains also two elements: (being) *on* (the table) and *crash* (off). From one among the possible architectures of the machine follows that $p(on/g) = 1$, $p(on/t) = 1$, $p(on/s) = 0$, $p(crash/g) = 0$, $p(crash/t) = 0$, $p(crash/s) = 1$, and that $u(on) = 1$ and $u(crash) = 0$. The values of the expected utility *e* of the actions *g*, *t* and *s*, resp. will be $e(g) = 1$, $e(t) = 1$, and $e(s) = 0$.

Because of the specificity of functions appearing in the previous description, the LADYBUG represents an instance of an llr-agent.

We conjecture that the low-level rationality is the maximal level of rationality, which may be achieved through subsuming purely reactive components.

Proceeding from the bottom of the reactive agents to the deliberative ones e.g. [53] suggests a new architecture based on the hypothesis that deliberation may be replaced (at least in some of its restricted forms) by the so called *imaginated interactions* – by pure reactivity of an agent to the remembered virtualized form of the environment in which it has been already situated. Stein describes an experimental mobile robot designed using this principle, a derivative of the robot *Toto* [39] – the *MetaToto*. Interactive abilities of the MetaToto include exploration of an office-like environment and goal-directed navigation within a previously explored and suitably remembered (in the form of a specific map) environment. Imagination enables to the robot to read and make use of the map allowing it to reason about an unfamiliar environment. In the following three sections we outline a way in which the phenomenon of low-level rationality can be analyzed in a framework based on the theory of grammars and grammar systems.

For an arbitrary agent to be active in an environment means to perform a sequence of acts. Each such sequence may be labeled by the corresponding sequence of symbols denoting the particular elementary acts performed by the agent.

From the language-theoretic point of view, the set of symbols denoting acts from A of an agent **A** may be understood as a finite alphabet, and a behavior **A** (the set of sequences of acts performed by this agent) can be considered as a language L_A over that alphabet. Formally, $L_A \subseteq A^*$, where A^* states for the set of all (finite) sequences (including the empty sequence) defined from the elements of A with respect of the binary operation of concatenation of (strings of) symbols. Thus A^* states, in fact, for all possible sequences (including the empty one) which may be formed from the acts performable by **A**, L_A is the set of all sequences, which can be effectively generated by **A** which is situated in an appropriate environment. If **A** is a *procedurally* rational agent, then there is some mechanism of **A** for selecting only a subset $L_{ratA} \subseteq L_A$ of all *rational behaviors* of **A**. If **A** is a substantively rational agent, then **A** is able to produce *only* behaviors from L_{ratA}, so $L_{ratA} = L_A$. Using this terminology, the question on the use of purely reactive agents, which behave without any internal representation of their environments, seems to be equivalent to that about the power of the substantive rationality.

From the definition of the expected utility function follows that the behavior L_A of an llr-agent **A** has the following property:

For arbitrary i, j $(1 \leq i \leq k; 1 \leq j \leq k)$, if $a_1 \ldots a_i, \ldots a_j \ldots a_k \in L_A$ then $a_1 \ldots a_j, \ldots a_i \ldots a_k \in L_A$. There exists an infinite class of infinite languages, which satisfy this property. Let us denote this class by $\mathbf{L_{INF}}$. Then, if A is an llr-agent then the behavior $\mathrm{L}_{\mathbf{A}}$ of **A** belongs to $\mathbf{L_{INF}}$ (is an infinite language). For the proof see (Kelemen 1996).

The adaptive behavior of the LADYBUG from the previous example can be (approximately) described by the set of all finite sequences of acts g (executable because of the physical limitations of the table maximally k-times) and t (executable because of the same reason maximally l-times): So, we have $L_{LADYBUG} = \{(g^m c^n g^r)^+ : 1 \leq m \leq k; 1 \leq n \leq l; 1 \leq r \leq k\}$.

This behavior can be defined by a formal grammar $G_{LADYBUG} = (N, T, P, S)$ where $N = \{S, A_1 .. A_k, B_1 ... B_k\}$ is the set of non-terminal symbols, $T = \{g, t, s\}$ is the set of terminal symbols, P is the set of productions containing the following rules: $S \rightarrow gA_1$, $\ldots A_{k-1} \rightarrow g A_k \mid t B_1$, $\mathrm{A_k} \rightarrow B_1$, $B_1 \rightarrow t A_2 \mid S \mid t$, $B_2 \rightarrow A_{k-1} B_3 \mid \mathrm{S} \mid t$, $\ldots B_k \rightarrow S$, and S is the starting non-terminal of the grammar G_{LADYBUG}.
The decision-theoretic model of the LADYBUG represents a centralized generative model of its rational behavior. It does not reflect the architectural principle applied in

construction of the LADYBUG considered as a totally decentralized set of two independent, autonomous, fully reactive (i.e. working without executing any recurrent computation) components acting in a shared environment.

We mentioned already that the LADYBUG is set up from two functionally separable mechanical parts, GO and TUR. Realize now that neither of these parts has any level of rationality – both of them (under suitable conditions, see below) crash off the table. However, subsuming them into an agent they generate a rational behavior. What are the crucial characteristics of such architecture?

First, the architecture enables the use of the ladybug's real environment as its own representation. Second, the architecture enables the elimination of any deliberation. The deliberation is "hardwired" in the system through subsuming of the reactive activities of the parts GO and TUR.

Let the behaviors of the two parts are $L_{GO} = \{g^m s_1 : 0 \leq m \leq k\}$ and $L_{TUR} = \{t^n s_2 : 0 \leq n \leq l\}$. Both of behaviors are finite languages because in all cases the strings of actions of the components lead (after executing a number of steps g or t) inevitably to reach the state s_1 or s_2 (by crashing the part GO or TUR off the table; let us suppose a slightly idealized condition where the table is quite small, so that the "infinite" rotation TUR is eliminated, that there are no obstacles on the table, no problems with parts' energy income, etc.). Trying to describe parts GO and TUR as llr-agents in the decision-theoretic framework we have:

Description of the part GO:

Acts: $A_{GO} = \{g, s_1\}$

States: $O_{GO} = \{on, crash\}$

Utilities: $u(on)=1$, $u(crash) = 0$

Description of the part TUR:

Acts: $A_{TUR} = \{t, s_2\}$

States: $O_{TUR} = \{on, crash\}$

Utilities: $u(on) = 1$, $u(crash) = 0$

In both of cases we have serious troubles with defining the beliefs: accepting the previous idealizations we may easily realize, that if the parts GO or TUR roll towards the table's edge, they necessarily crash (so, that $p(crash/g) = p(crash/t) = 1$). Thereupon, the parts GO and TUR are not llr-agents in the sense of our previous definition. However, both of the behaviors can be described by corresponding formal grammars:

The formal grammar for the part GO, G_{GO}, consists of the set $N_1 = \{A_0, A_1, \dots A_k\}$ of non-terminal symbols, the set $T_1 = \{g, s_1\}$ of terminals, the set P_1 containing the productions $A_0 \rightarrow gA_1$, $A_1 \rightarrow gA_2 \mid s_1$, ... $A_{(k-1)} \rightarrow gA_k \mid s_1$, $A_k \rightarrow s_1$, and the starting non-terminal A_0.

The grammar G_{TUR} consists of the set $N_2 = \{B_0, B_1, \dots B_k\}$ of non-terminal symbols, the set $T_1 = \{t, s_2\}$ of terminals, the set P_2 containing the productions $B_0 \rightarrow tB_1$, $B_1 \rightarrow tB_2 \mid s_2$, ... $B_{(l-1)} \rightarrow gB_l \mid s_2$, $B_l \rightarrow s_2$, and the starting non-terminal B_0.

We have two regular grammars generating finite languages L_{GO} and L_{TUR}, resp. Now, in G_{GO} and G_{TUR}, instead of s_1 and s_2 we will write B_0 and A_0, resp. Then the colony formed from G_{GO} and G_{TUR} generates, starting from A_0, the language $L_{LADYBUG} = \{(g^m c^n g^r)^+ : 1 \leq m \leq k; 1 \leq n \leq l; 1 \leq r \leq k\}$.

Concerning the rationality of colonies of agents, in [21] is proved that the class of llr-agents with behaviors generated by colonies is infinite. This shows, that at least the low-level rationality may appear as an emergent effect of unsupervised individual behaviors of a finite number of autonomous purely reactive non-rational components. It is also clear, that this rationality is substantive in its nature, because there are no decision-making procedures performed by some component(s) in colonies.

A possible criticism of the approach to rationality presented in this Section may start from realizing the remote credibility of the presented formal, and hence simplified model. Our only intention has been, however, to illustrate the possibility of emergence of a low level of rationality in systems set up from non-rational reactive components. Many important problems connected with the real complex phenomenon of rationality remains untouched, of course. We admit that it is caused by the principal bounds of the power of the used simple formal framework. If it is true, then this Section contributed to identification of a particular level of rationality (adaptiveness), maybe the lowest one, and may be closed with the hypothesis that the substantive rationality of agents coincides with (some variants of) their adaptiveness, with the lowest level of their rationality.

References

1. Arkin, R. C. (Ed.): *Robot Colonies*. Kluwer, Boston, Mass., 1997
2. Arkin, R. C.: *Behavior-Based Robotics*. The MIT Press, Cambridge, Mass., 1998
3. Bechtel, W., Richardson, W. C.: *Discovering Complexity*. Princeton University Press: Princeton, NJ, 1993
4. Baník, I.: Colonies with positions. *Computers and Artificial Intelligence* **15** (1996) 141-154
5. Baník, I.: Colonies as systems of Turing machines without states. *Journal of Automata, Languages and Combinatorics* **1** (1996) 81-96
6. Brooks, R. A.: *Cambrian Intelligence*. The MIT Press, Cambridge, Mass., 1999
7. Connell, J. H.: *Minimalist Mobile Robotics – A Colony Architecture for a Mobile Robot*. Academic Press, New York, 1990

8. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Paun, Gh.: *Grammar Systems – A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach, London, 1994
9. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Paun, Gh.: Eco-grammar systems – a grammatical framework for lifelike interactions. *Artificial Life* **3** (1997) 1-28
10. Dassow, J., Kelemen, J., Paun, Gh.: On parallelism in colonies. *Cybernetics and Systems* **24** (1993) 37-49
11. Dassow, J., Paun, Gh., Rozenberg, G.: Grammar systems. In: *Handbook of Formal Languages, vol. 2.* (G. Rozenberg and A. Salomaa, eds.) Springer-Verlag, Berlin, 1997, pp. 155-214
12. Doyle, J.: *Artificial Intelligence and Rational Self-Government (Technical Report CMU-CS-88-124)*, Computer Science Department, Carnegie Mellon University, Pittsburgh, Penn. 1988
13. Epstein, J. M., Axtell, R.: *Growing Artificial Societies – Social Science from the Bottom Up*. The MIT Press, Cambridge, Mass., 1996
14. Ferber, J.: *Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Harlow, 1999
15. Forrest, S.: Emergent computation – self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. In: *Emergent Computation* (S. Forrest, Ed.). The MIT Press, Cambridge, Mass., 1991, pp. 1-11
16. Freund, R., Kelemenová, A. (eds.): *Proc. Intern. Workshop Grammar Systems 2000*. Silesian University, Opava, 2000
17. Freund, R., Kelemen, J., Paun, Gh.: *Grammar Systems, Abstract Families of Languages, and the Emergence* (in preparation)
18. Gašo, J.: Unreliable colonies – the sequential case. *Journal of Automata, Languages and Combinatorics* **5** (2000) 31-44
19. Gašo, J.: Unreliable colonies – the parallel case. In: *Proc. International Workshop Grammar Systems 2000* (R. Freund and A. Kelemenová, eds.). Silesian University, Opava, 2000, pp. 189-201
20. Holland, J. H.: *Emergence – from Chaos to Order*. Addison-Wesley, Reading, Mass., 1998
21. Kelemen, J.: A note on achieving low-level rationality from pure reactivity. *Journal of Experimental & Theoretical Artificial Intelligence* **8** (1996) 121-127
22. Kelemen, J.: Colonies as models of reactive systems. In: *New Trends in Formal Languages* (Gh. Paun and A. Salomaa, eds.). Springer-Verlag, Berlin, 1997, pp. 220-235
23. Kelemen, J.: Colonies – grammars of reactive systems. In: *Artificial Intelligence and Information-Control Systems of Robots 97* (I. Plander, Ed.). World Scientific, Singapore, 1997, pp. 27-40
24. Kelemen, J.: Colonies – a theory of reactive agents. In: *Proc. MFCS 98 Satellite Workshop on Grammar Systems* (A. Kelemenová, Ed.). Silesian University, Opava, 1998, pp. 7-38
25. Kelemen, J. (guest editor): *Grammars* **1** (1999) 181-292
26. Kelemen, J.: On post-modularity and emergence from grammar-theoretic point of view. In: *Quo Vadis Computational Intelligence?* (P. Sinčák, J. Vaščák, eds.). Physica-Verlag, Heidelberg, 2000, pp. 342-352
27. Kelemen, J., Kelemenová, A.: A grammar-theoretic treatment of multiagent systems. *Cybernetics and Systems* **23** (1992) 621-633
28. Kelemen, J., Kelemenová, A. (guest editors): *Journal of Automata, Languages and Combinatorics* **5** (2000) 1-78
29. Kelemen, J., Kelemenová, A., Martín-Vide, C., Mitrana, V.: Colonies with bounded lifetime components. *Theoretical Computer Science* **244** (2000) 289-298

30. Kelemen, J., Kelemenová, A., Mitrana, V.: Neo-modularity and colonies. In: *Where Mathematics, Computer Science, Linguistics and Biology Meet* (C. Martín-Vide and V. Mitrana, eds.). Kluwer, Dordrecht, 2001, pp. 63-74
31. Kelemen, J., Paun, Gh. (guest editors): *Computers and Artificial Intelligence* **15** (1996) 105-272
32. Kelemen, J., Paun, Gh.: Robustness of decentralized knowledge systems – a grammar-theoretic view. *Journal of Experimental and Theoretical Artificial Intelligence* **12** (2000) 91-100
33. Kelemenová, A. (editor): *Proceedings of the MFCS '98 Satellite Workshop on Grammar Systems*. Silesian University, Opava, 1998
34. Kelemenová, A.: Timing in colonies. In: *Grammatical Models of Multi-Agent Systems*. (Gh. Paun and A. Salomaa, eds.). Gordon and Breach, London, 1999, pp. 136-143
35. Kelemenová, A., Csuhaj-Varjú, E.: Languages of colonies. *Theoretical Computer Science* **134** (1994) 119-130
36. Langton, Ch. (Ed.): *Artificial Life – An Overview*. The MIT Press, Cambridge, Mass., 1995
37. Martín-Vide, C., Paun, Gh.: PM-colonies. *Computers and Artificial Intelligence* **17** (1998) 553-582
38. Martín-Vide, C., Paun, Gh.: New topics in colonies theory. *Grammars* **1** (1999) 209-223
39. Mataric, M. J.: Integration of representation into goal-driven behavior-based robots. *IEEE Journal of Robotics and Automation* **8** (1992) 304-312
40. McClamrock, R.: *Existential Cognition*. The University of Chicago Press: Chicago, Ill., 1995
41. Paun, Gh.: On the generative power of colonies. *Kybernetika* **31** (1995) 83-97
42. Paun, Gh. (editor): *Artificial Life – Grammatical Models*. Black Sea University Press, Bucharest, 1995
43. Paun, Gh., Salomaa, A. (editors): *Grammatical Models of Multi-Agent Systems*. Gordon and Breach, London, 1999
44. Pollock, J. L.: New foundations for practical reasoning. *Mind and Machines* **2** (1992) 113-144
45. Pylyshyn, Z.: The role of competence theories in cognitive psychology. *Journal of Psycholinguistic Research* **2** (1973) 21-50
46. Ronald, E. M. A., Sipper, M., Capcarrére, M. S.: Design, observation, surprise! A test of emergence. *Artificial Life* **5** (1999) 225-239
47. Russell, S., Norvig, P.: *Artificial Intelligence – A Modern Approach*. Prentice Hall: Upper Saddle River, NJ 1995
48. Russell, S., Subramanian, D.: Provably bounded optimal agents. In: *Proc. 13th IJCAI*, Morgan Kaufmann, Palo Alto, Cal. 1993
49. Simon, H. A.: Rationality as process and as product. *American Economic Review* **68** (1978) 1-16
50. Simon, H. A.: *The Sciences of the Artificial (2nd Edition)*. The MIT Press, Cambridge, Mass. 1982
51. Sosík, P.: Parallel accepting colonies and neural networks. In: *Grammatical Models of Multi-Agent Systems*. (Gh. Paun and A. Salomaa, eds.) Gordon and Breach, London, 1999, pp. 144-156
52. Sosík, P., Štýbnar, L.: Grammatical inference of colonies. In: *New Trends in Formal Languages* (Gh. Paun, A. Salomaa, eds.). Springer-Verlag, Berlin, 1997, pp. 236-245
53. Stein, L. A.: Imagination and situated cognition. *Journal of Experimental and Theoretical Artificial Intelligence* **6** (1994) 393-407
54. Stein, L. A.: Post-modular systems – architectural principles for cognitive robotics. *Cybernetics and Systems* **28** (1997) 471-487

Emotions and Agents

Paolo Petta[1] and Robert Trappl[2,1]

[1] Austrian Research Institute for Artificial Intelligence,
Schottengasse 3,
A 1010 Vienna, Austria (EU)
paolo@oefai.at
http://www.oefai.at/oefai/agents/

[2] Dept. of Medical Cybernetics and Artificial Intelligence,
Freyung 6/II,
A 1010 Vienna, Austria (EU)
robert@ai.univie.ac.at
http://www.ai.univie.ac.at/∼robert

Abstract. The encounter between emotion research and agent-based technology is multifaceted. One the one hand, results from emotion research start to serve as role model from nature, providing inspirations for technical design criteria for individual agents at the micro level and agent groups and societies at the macro level as well as the sophisticated linkages in between them. On the other hand, they are of immediate impact in important aspects of human-agent interaction and effective social cooperation between humans and conversational interfaces. In this broad survey, we offer an appetising selection of results from different areas of emotion research.

1 Introduction

At the latest with the coining of the term "Affective Computing" by Rosalind Picard [57] and its subsequent rapid diffusion the natural phenomenon of emotions has become established as a topic of interest for a wide range of computing issues, including agent-based approaches in particular. Here, emotions on the one hand serve as natural role model, providing inspirations for technical design criteria for individual agents at the micro level, agent groups and societies at the macro level, and sophisticated linkages in between them. Furthermore, they are of immediate relevance for important aspects of human-agent interaction, which do not apply to embodied agents only (e.g., [10]), but are also of paramount importance for effective social cooperation between humans and conversational interfaces to information agents (e.g., [59,5]). Finally, in all these and many further aspects the topic of emotions serves as a busily humming hub intensifying interdisciplinary communication between computer scientists and an ever growing number of scientific disciplines [74,76] including artificial life and cognitive robotics [12,77,25], biology [35]; cognitive science [15,16]; mathematics [3], neurology [17,18,41,62], philosophy [71,72], psychology [43], and sociology [38], to name a few.

As will be illustrated to some detail, in natural individuals and societies emotions play an instrumental role in *relational* activities. In other words, they are crucial for the successful *interaction* of autonomous individuals with *their* (subjective, individual) *environment* (which usually also comprises themselves as well as further similar individuals). Affective channels of communication provide continuously updated situational and contextual information of an individual's or a group's state, including clues about current tendencies to act (e.g., whether intending to approach or retreat, whether willing to pay attention, whether willing to give in or oppose). By nature, humans are highly attuned to the (in part unconditional) pick up of this kind of (partly unconditionally published) information, which forms an integral part of the coordination of social activities. To an interface agent designer, this state of affairs could be marketed as the availability of an ubiquitous ("100% installed!"), efficient and well-trained real-time information channel whose dynamics ought at least to be modeled passively, so as to be able to observe and understand what is going on in the human partner(s), in order to adapt accordingly.

These channels and the natural processes utilising them, however, gain an even greater importance with long(er)-lasting, persistent tasks and relationships. This is because any situated entity (such as you, the reader, but also any agent "thrown" into its real-time environment) simply does not have the options of *not* behaving ("resting" or "pausing" manifestly are themselves behaviours, as is shutting one's eyes) and, even more, of not behaving *expressively.* In other words, *any* behaviour is readily characterised in the eyes of an observer in emotional terms. It follows immediately that paying attention to these facets of emotions—not only in an descriptive observational model of human users, but in a prescriptive model capable of influencing the agent's own behaviour—is not at all a matter of choice for a user interface designer (see section 3, as well as the chapters on Virtual Sales Agents and Intelligent Interface Agents in this volume): explicit attention to the emotional aspects of computer interaction is necessary in order to avoid degrading the user's experience by generating unnatural and disconcerting behaviours! But also engineers tackling issues of internal coordination in artificial multi-agent systems might become intrigued and motivated to try to find out more about which powerful mechanisms are at work in natural societies and assess whether they may serve as useful inspirations for innovations in their own domain.

The last aim of research into emotions and agents to be mentioned here can be viewed simultaneously as both, the one holding the highest innovative potential by forcefully pointing the way towards progress in the design of situated autonomous agents, and the most controversial in terms of actual achievability. Beyond the two previously mentioned steps of merely modelling some (e.g., expert system like) description of emotions—as they are taken to occur in humans, and as they may occur in interactions of humans and agents (or even between agents endowed with, say, emotional conversation policies)—lies the question of the possibility of implementing agents that *have* emotions. The authors' views in this respect coincide largely with the ones expressed e.g. in [69,70], [46], [55], [71,72], or [58]: at least to the extent that, as in psychology, renewed interest in emotions reflects the final overcoming of the mind-body split (which in the present context might be associated e.g. to the "grounding problem" in artificial intelligence); to the extent that emotions are taken to be that which plays a fundamental role in the interface between impenetrable (unconscious, irreflexive) and inspectable (conscious, reflexive) aspects of cognition; to the extent that the notion of what is good and what is bad for the individual is taken as a necessary condition of autonomous existence [56]; to the extent that the notion of emotions is associated to what occurs within an autonomous entity when perceiving (i.e., appraising a stimulus event that is significant for its relationship to the world [29]), viz. (re-)coordinating itself with its environment [15], it seems unlikely that appreciable intelligent autonomous behaviour could be achieved without emotions.

2 Results from Emotion Research

From an observer's point of view, *emotion* appears as a hypothesis to explain behaviour that has neither sufficient nor adequate external purpose or reason, behaviour characterised by an apparent "surplus" not needed for the end results, such as superfluous emphasis in speed and scope of movement. This explanation is straightforwardly sought within the subject observed. Bridging from an outside observer to subjective experience stand physiological manifestations of emotions, such as changes in temperature, blood pressure, and perspiration. Finally, there are the private, internal aspects of emotional experience, which are roughly characterised by two features shared across them: first, an evaluative connotation (ratings along scales such as good–bad or weak–strong) and second, subjective reference ("*I* feel...", "*I* am in the state of...") [29]. Similarly, the following overview sets out from the point of view of an external observer, as typical in ethology. Intra-personal aspects concerning the neurophysiology of single individuals are discussed next, followed by more abstract observations in terms of cognitive models that take into account inter-personal aspects of emotions. We conclude this section with some remarks about current social models of emotional phenomena.

2.1 Emotions as Natural Behaviour

Assuming that emotions are a natural phenomenon occurring in situated real-time systems, what functionalities are emotions then seen to contribute to the survivability and fitness of animals and animal societies, and by which means? Answers usually involve the notion of quantifiable fixed action patterns (FAPs): recognisable, repeated patterns within what at first would appear to be a continuous flow of action [35]. The hypothesis of the existence of such discrete behaviour modules also in humans found support, in particular in the domain of facial expressions and related body language, and for situations requiring fast action such as in fleeing or fighting. Study of FAPs in animals and humans has uncovered general properties that contribute to solve a range of difficult challenges and provide essential features such as homeostasis (see also the subsection about the Somatic Marker hypothesis, below) by virtue of such sophisticated features as "declaratively situated" goals (i.e., defined in terms of a target perception to be achieved and maintained); orienting components providing corrective movements to counter environmental perturbations; timing properties combining initial persistence with temporal boundedness; context dependency for the effectiveness of releasing stimuli; and modifiability of the repertory of perceptors and behaviours.

Disposing of a nontrivial number of different FAPs (and related perceptual releasers, etc.) brings about the problem of an optimal action selection strategy. Emotional states in animals are commonly defined by the group of FAPs that are potentiated (i.e., share an increased probability of being enacted) when a given stimulus is presented. Similar to the timing properties in FAPs, an emotional state preserves the potentiating of the defining FAPs over a time interval that can exceed by far the duration of the actual confrontation with the eliciting stimulus. This persistence of bias towards a one-time decision about probably relevant FAPs reflects the recurrent existence of environmental states of more or less protracted stability. Mapping of environmental information to a probabilistic bias towards a group of FAPs selected by the combination of evolution and individual adaptation is a plausible way to exploit best the capabilities of these two learning variants while deferring the actual selection of action to the latest possible moment. The complexity of certain contexts that determine the effectiveness of associated stimuli and the corresponding existence of specialised emotional states indicates high cognitive demands involved, whether in sheer number of available FAPs or sophistication of required information processing capabilities. The next subsection consequently takes us into the realm of neurophysiological research.

2.2 The Neuro-physiological Substrate of Emotions

The previous subsection has shown how the introduction of emotional states to control an inferior layer of fixed action patterns could contribute to improve the (necessarily always real-time!) performance of an organism equipped with larger behavioural and perceptual capabilities in an environment posing increasingly sophisticated demands—e.g. with the advent of social cooperation. The link from the faculty to associate complex stimuli worth reacting to with emotional states and the link from emotional states to higher cognitive demands suggests how emotional performance may form an evolutionary incentive towards the development of cognitive capacities of increasing sophistication.

Reward and Punishment A radically clear formulation of this hypothesis is provided by Edmund T. Rolls [62]. He views emotions as states elicited by instrumental reinforcing stimuli, i.e., rewards (anything worth working for) and punishments (all it is worth investing effort to escape or avoid), and shows this working assumption to be sufficient to explain a wider range of emotional phenomena.

This principle allows a characterisation of required structures to implement it and the assessment of whether there exist elements of the known organisation of the brain that might reasonably map to this hypothesised design. The central role of reward and punishment evaluation systems is crucial for the explanation of how the construction of complex systems that produce appropriate but flexible behaviour to increase their fitness might be possible: rewards and punishments are specified as goals rather than particular behavioural patterns of responses, allowing for an open range of possible behavioural strategies to deal with them (cf. the characteristics of fixed action patterns in the previous subsection). Given some stimuli that are unlearned primary reinforcers (e.g. pain, or the taste of food), others may turn secondary positive reinforcers (rewards) by association, increasing the probability of emission of a response on which they are contingent. Analogously, negative reinforcers (punishers) decrease the probability responses. Converse reinforcement contingencies produce opposite behavioural effects: the omission or termination of a positive reinforcer decreases the probability of responses, while responses followed by the omission or termination of a negative reinforcer increase in probability. According to this analysis, emotions can thus be produced by the delivery, omission, or termination of stimuli.

The list of functions of emotions supported by Rolls' model is compatible with the one given at the end of the previous subsection, and includes the social aspects of communication via expressive behaviour and bonding as well as impacts on cognitive evaluation of events and memories and cerebral plasticity (episodic memory, biasing of recall, establishing of representations and analysers). Rolls' model of action selection expands on the one presented in subsection 2.1; his dual pathway analysis sees the "lower, implicit" system adapting to the overall distribution of opportunities under consideration of directly available rewards only, while the "upper, explicit" one allows for a situation-specific tailored implementation of one-off plans, occasionally even under deferral of achievement of higher rewards. Importantly, Rolls proposes adaptive influences of either system on the other.

The Anatomy of Fear In contrast to the principled approach pursued by Rolls, Joseph LeDoux [41] generalises from empirical findings. He suggests the existence of brain circuits forming a "low road" and performing "quick and dirty" unconscious appraisals well in advance of the more elaborated evaluations that take place via a different "high road" and eventually register in consciousness leading to awareness of the bodily responses already initiated by the first system. The speed advantage of the low road enables the individual to react rapidly to danger situations without having to evaluate alternatives; however, one may also react wrongly to inoffensive situations because the low road is only capable of a coarse differentiation of stimuli. To contain this problem, the high road can exert some limited regulatory functions on the lower one. LeDoux proposes a structural and functional architecture of the brain that accounts for the way emotion and cognition interact in determining behaviour. The brain is seen as an assembly of many specialised systems carrying sets of specific roles. General brain functions such as emotion come about by the interplay of several brain systems, some of which—including the amygdala as a distinct emotion system—possess their own implicit memory.

Similarly to Rolls, LeDoux also sustains emotion's supporting role in instrumental learning; the arousal brain system is seen to contribute to the persistence of initiated emotional state. By inducing stress, emotion may influence the formation of flashbulb memories. These are very detailed but not very accurate, as the memorised details depend on the subjective relevance within the particular stressful situation. High stress can cause severe explicit memory deficits but improves the formation of implicit memories by the amygdala; a subsequent presentation of a stressful stimulus will thereby lead to high arousal that cannot be explained. In this way, stress may act as a switch between situations where deliberation is appropriate, and stressful circumstances where automatic decisions ensure timely response. Like Rolls, LeDoux asserts the possibility of having emotional reactions caused by retrieval of particular memories. In agreement with certain multi-level theories discussed below, LeDoux provides an anatomical model of how results of early stimulus evaluation can control attention.

Somatic Markers and Consciousness The publications by António Damásio rank among the most influential in this area of research. Damásio asserts an intimate connection of emotion's processes to homeostasis: emotions are biologically determined processes that have a regulation role that leads into a profitable situation for the organism. In [17], *primary emotions* are defined as processes relying on innate detectors of the single or simultaneous presence of certain features of stimuli in the world or in one's body and causing changes in bodily state as well as cognitive processing. Beyond accomplishing useful goals such as fast hiding from a predator or display of anger towards competitors, the ensuing conscious *feeling* of the primary emotion offers flexibility of response based on the particular history of interactions with the environment. The formation of systematic connections between categories of objects and situations on the one hand and primary emotions on the other leads to the triggering of *secondary emotions* which reflect the personal experiences of the individual.

A special instance of feelings generated from these secondary emotions, namely those that have been connected by learning to predicted future outcomes of certain scenarios, are called *somatic markers*. Somatic markers are taken to function as a biasing device in that they substantially prune the search space of possible courses of actions by eliminating paths of likely unfavourable outcome. *Jointly* with subsequent deliberate rational decision making, somatic markers thus are likely to increase the accuracy and efficiency of the decision process, while their absence also has an explicit, negative, effect. An important function of somatic markers is seen in their support in overcoming short term horizon effects, as in the choice of actions whose immediate consequences are negative, but which generate positive future outcomes (cf. also Robert Frank's notion of "moral sentiments" as devices to commit a person to act contrary to immediate self-interest in [28]). While the somatic marker theory in [17]the underscores the involvement of the whole body of an organism in the formation and processing of emotions, more recent investigations [18] address the importance emotion may have in the creation of consciousness, and how perception of emotions may not be possible without consciousness.

The brief selection covered in this subsection could not possibly but provide a first idea of the impressive results obtained in neuro-physiological domains, until recently considered to be necessarily well recessed from any immediate relevance for agent applications. Examples of current research following up on these architectural designs and similar efforts can be found e.g. in [15,12,11,77], and [25].

2.3 Emotion and Cognition

Perhaps one of the most prominent examples of current cognitive theories of emotions, the cognitive appraisal theory of emotions [2,29,65,66] proposes the view that emotions serve an adaptive purpose, if not at all instantiations. The characteristics of individuals to be subserved by the emotions include [33]: *autonomy*—the capability to exert and defend control over some processes with only indirect influence from the environment; achievement or maintenance of ultimate goals termed *concerns* using the autonomy the system is endowed with (including the very preservation and flexible management of autonomy in cooperation with other individuals); *social communication*: generation of and sensitivity to social signals using different modalities (e.g., visual, acoustical, or textual); and coping with *resource limitations* in terms of information acquisition, processing, and generation capabilities. Typical characteristics of the environment of relevance for the present investigation include: *finiteness of resources* that may be used to satisfy any active goals of the individuals; *provision of signals* indicating the possible presence of opportunities to meet or maintain current goals or of detrimental circumstances; (apparent) *non-determinism* w.r.t. the occurrence of events, the outcome of initiated actions, and the reliability of signal generation and reception—in particular, it cannot be anticipated with certainty, as in the context of affective interaction, where satisfaction of or threats to concerns is fundamentally probabilistic. The environment is a *social* one, populated by individuals that may compete for resources (including—temporary—control over others), may cooperate in the achievement of joint activities, and are in turn sensitive to symbolic social signals in addition to physical events or actions. Finally, a complete lifeworld analysis also covers the conventions and invariants maintained by agents throughout their activities [1]. Beyond ensuring homeostasis of at least those concerns necessary for survival, in the particular context of affective interaction this regards the compliance with social norms and the living up to individual standards and values, at least most of the time [73]. A rather elementary corollary to this analysis is the observation that the degree to which any of these characteristics are actually implemented in any artificial system matters only to the degree to which this can or actually is called for by the users interacting with the system, point in case the successes of systems such as ELIZA [78] or JULIA [27] (cf. also the analysis given in [61]).

With reference to the fundamental properties of emotional systems presented above, the following functions of emotions are identified by cognitive appraisal theorists: signalling of events of subjective relevance; detection of difficulties in problem solving; provision of temporary goals along with associated internal resource management; and parallel pursuit of all personal concerns at all times. Supporters of appraisal theory emphasise that emotion cannot possibly occur without both cognition and motivation, as emotion is always a response to *meaning*, including the implications of a transaction for one's personal goals, regardless of how that meaning was achieved; without a goal at stake there would be no emotion.

A rough basic model of the emotion process is seen to run as follows: the confrontation of the current state of the set of concerns of an individual with a given emotion eliciting event leads to the appraisal of its significance for (changes of) the individual's relationship to its environment. This construed interpretation of the event is captured in a theoretical abstraction termed situational meaning structure, which consists of components determined at different times (cf. the explications arguing for the existence of different levels of analysis performing at different speeds given in the previous sections). A first assessment regards what the event at hand can do or offer; a second one regards the estimation of one's coping potential with respect to this interpretation—the assessment of the availability and possible applicability of any coping action according to contextual circumstances. This leads to the generation of action readiness change, expressed as an *action tendency* that may or may not lead overt action. Action tendencies correspond to what in ethology is called the activation of behaviour systems which address functionally equivalent aims that tend to co-occur under the same circumstances (cf. 2.1). Finally, the notion of regulation encompasses all related processes that serve to amplify, attenuate or maintain the strength of emotional reactions by influencing any of the aforementioned components [34]. It can be only noted here that this model also allows for a straightforward consistent coverage of personality as consistent reactive bias within the fringe of functionality [47].

Cognitions for Emotions The debate about the minimal cognitive prerequisites for emotion has generated much attention, in spite of the fact that it quickly settled onto semantic issues, where two basic topics need to be clearly distinguished: the target of appraisal theory predictions; and the meaning of the terms cognition, evaluation, or appraisal [65]. With respect to the further, it has to be stressed that appraisal theory focuses on the reaction to significant stimulus events that change the organism-environment relationship. The target of the prediction is a fully instantiated emotion, in the sense of the determination of a sufficiently well established situational meaning structure. Current appraisal theories develop an incremental approach, starting from the basic establishment of fundamental action tendencies that then interface to the wide-open range of domain-dependent individual and unique emotional experiences that are equipped with all further elements of subjective experience. With respect to the latter, much of the debate depends on the definition of the notion of cognition (often mixed up with that of consciousness) or on the connotations of the concepts of appraisal or evaluation. An increasing number of researchers also support an overarching notion of appraisal that encompasses virtually every type of information processing, acknowledging the fundamental fact that it is not the objective nature of a stimulus but the organism's evaluation of it that determines the nature of the ensuing emotion. In agreement with the tenets of the "Nouvelle AI" movement [56] it is asserted that even a completely automatic, reflexive defence reaction of an individual constitutes an intrinsic assessment of the noxiousness of the stimulus (even though it may not necessarily produce a full-fledged emotion).

A number of theories link the appraisal mechanism directly to other components of the emotion process. These generally share the view that appraisal outcomes should cause an appropriate adaptive reaction in the other emotion process components. Klaus Scherer [64] has presented rather detailed prediction tables for the presumed effects of appraisal outcomes on facial and vocal expression, physiological responses and behaviour tendencies. Nico Frijda [29,31] has mainly focused on the relationship between appraisal outcomes and action tendencies and has demonstrated the existence of appraisal-action tendency links; similarly, [50] and [63] have also suggested direct links between appraisal categories and response patterns.

Basic Emotions? Across virtually every current theory of emotion there is a consensus regarding the existence of fundamental, primitive and "pre-designed" mechanisms that already bring about the flexible reactive connectivity between internal or external events with low stimulus specificity and dynamically varying personal goals that is distinguishing of emotions. This characteristic complements and exceeds the capabilities of either reflexes (highly specific stimulus and low response flexibility) and physiological drives (specific stimulus and moderate response flexibility). From the theoretical point of view it is then a fallacy to simply equate this commonly described state of affairs with the existence of some fixed number of "basic emotions". The "basic emotions" approach quickly runs into problems most poignantly apparent with respect to the usually accompanying notion of "mixed", or "compound" emotions, which are supposed to result from combinations of basic emotions and to suffice for the generation of any desired or observed emotional state. Recently, the originator of the term "basic emotions", Paul Ekman, himself pinpointed the fact that he does no longer allow for "non-basic" emotions; all the emotions sharing his defining characteristics being now referred to as "basic" [20]. We think it is apparent how available alternatives such as Damásio's account of primary emotions described above, as well as various others that cannot be covered in this short paper, should be preferable for almost all purposes [50,40]. Among the few exceptions remain the typically patchy or idiosyncratic approaches characteristic of the shallow and limited domains currently still widespread in entertainment applications of agent technology. If an artificial, flashy and coarse-grained highlighting of "emotional" states that should differ from one another as much as possible is the goal, then the application of any of the classical lists of 5–7 "basic emotions" (e.g., happiness, surprise, anger, fear, sadness, and disgust) would seem to be most appropriate (see also the references given in section 3).

Multi-level Theories of Emotion Progress in cognitive appraisal theory research is developing along different lines. On one hand, the characteristics of single identified entities are being fleshed out at an increasing level of detail. On the other hand, there is increased interest in the construction of process models that specify how the theoretically postulated functionalities can be brought about. Naturally, this latter development leads to an increased interaction with other strands of emotion research, from neurophysiology to affective computing, thereby contributing to the overall convergence in this area. Multi-level theories of cognition and emotion recognise qualitatively distinct kinds, or levels, of information and representation. They suggest that, in attempting to understand the emotional effects of events, it is helpful to consider separately the contributions from different kinds of information, and their interactions, rather than to lump them all together in some general concept of cognition. Multi-level theories treat both appraisals and perceptual features simply as information at different levels of abstraction. In this way, these theories allow to focus clearly on the central tasks of characterising those levels, and their relationship to emotion, rather than on arguing about the boundary conditions for the use of the term cognition.

2.4 Sociology

Early studies of emotion largely tended to focus on intra-personal aspects of emotion, mapping the determinants and characteristics of emotional response within the individual, one notable exception being formed by research on the interpersonal functions of facial expressions, beginning with the seminal publication of Charles Darwin [19]. More recently, results from sociology, ethology and anthropology have led to greater awareness of the ways that emotions construct and are constructed by cultural practices and institutions.

Keltner and Haidt [38] structure social functional research of emotions into *individual* (intrapersonal), *dyadic* (between pairs of individuals), *group* (a set of individuals that directly interact with some temporal continuity), and *cultural* (within a large group that shares beliefs, norms, and cultural models). While at the individual and dual level of analysis adaptationist arguments in agreement with evolutionary theory are widespread, work at the group and cultural levels of analysis focuses on emotions as cultural products constructed by individuals or groups in social contexts, linked to construals of the self, patterns of social hierarchy, language or socio-economic requirements. In this latter view, there is no equivalent to natural selection, rather, socially constructed emotions fit with social structures and other cultural facts in ways that make sense from an interpretative viewpoint (see e.g. [39]). Even so, some assumptions are commonly shared across all levels, namely that people are social by nature and meet the problems of survival in relationships; that emotions serve to co-ordinate social interactions and relationships to address those problems; that emotions are relatively automatic, involuntary, and rapid responses that help humans in different social relationships, often for their own benefit [28]; and that emotions are dynamic processes that mediate the individual's relation to a continually changing social environment. Building on these commonalities, social multi-level accounts are now starting to appear, in a way analogously to the integration of intra-organismical accounts in multi-level theories, as just portrayed.

3 Human-Agent Interaction

As we do not have space to expand on this topic, we refer the reader once more to related chapters in this volume. In particular, we have to leave aside almost all aspects regarding "design-time" emotions, i.e., artistic effects aimed at obviating the need to endow the participating agents with any substantial degree of emotion processing capabilities. These and the related discussion on emotion/personality vs. believability tradeoffs are covered at length e.g. in [60,22,44,74,23,45].

3.1 A Bayesian Sidestep

The following is a noteworthy example for the application of a technology known also from traditional expert system design that allows to sidestep many issues that currently cannot be tackled head-on by deep modeling. Researchers at Microsoft are investigating the application of Bayesian Networks, a formalism for representing networks of probabilistic causal interactions [37], to the modelling of emotion and personality [4,5,6]. The Bayesian model employed contains two internal variables each for emotional state (characterised in terms of Valence and Arousal) and for the personality profile (represented as a pair of Dominance=tendency to be controlling, and Friendliness=tendency to be friendly/sympathetic). These four nodes are treated as unobservable variables in the Bayesian formalism. This quite simple but descriptive model of the internal emotional state and personality type of an individual is related to behaviours that help to communicate that state to others via links connecting internal states to nodes representing aspects of behaviour judged to be influenced by that hidden state. Behaviour nodes currently represented include linguistic behaviour (e.g., word selection), vocal expression (base and variability of pitch, speed and energy of speech), posture, and facial expressions. The model furthermore includes temporal dependencies between current and previous values of the internal emotion variables over fixed timeslices of a few seconds duration.

Relying on the fundamental consistency of the facets of the natural phenomenon to be captured, this shallow approach is capable of offering some remarkable possibilities. For example, the same network can be used both to calculate the likely consequences of changes to their causal nodes, as well as to diagnose the likely causes of a collection of observed values at the dependent nodes; i.e., a single parametrised network can be used for both emotion recognition and simulation of emotional response. Another advantage is the ease of extending the range of information sources covered, given that only the single "universal currency" of probabilistic interdependencies is considered. Clearly, these advantages are bought at the expense of the acquisition of a complex numerical matrix that is difficult to maintain and adapt in a well-aimed manner.

3.2 Mining User Activity

As noted by Kia Höök, the active role of the users is not immediately affected if synthetic personal character assistants are implemented resorting to shallow tricks of the trade. A basic tenet of narrative theory in general, and discourse psychology in particular, is the notion of readers/spectators constantly striving for coherence in their understanding and experience of a given text [8]. Coherence is thus *constructed* by the reader who—under the influence of pragmatic parameters, such as the purpose of the interaction—unknowingly resorts to a huge library of tacit common knowledge based beliefs about the perceptual, physical and socio-cultural world.

3.3 Design Problems

A number of design problems stand out when considering the topical issue of Emotion and Agents in the context of human-agent interaction, but are rapidly gaining in relevance also in embodied multi-agent scenarios, where agents are equipped with virtual sensors. Agents have to *behave* intelligibly: rather than just acting with near-perfect rationality, their movements should convey a richer description of the whole *personal story* they are unfolding [67,51]. Relatedly, there are delicate timing issues both at the micro level (synchronisation of different modalities involved in the transmission of affective content) and at the macro level (the combined information must be made available at the right pace) [36].

Agents sometimes need to have interesting personalities so that their emotional behaviour becomes both comprehensible and interesting to the user. This however brings about the requirement of adequate interaction length and richness of context for the personality to be actually manifested, and raises the related issue of negotiation for interaction time, with the user as well as with other agents (that might contribute to a fuller rendering of existing personality traits). Finally, the range of believable—and probably also functional—emotional adaption of any given agent necessarily is limited. As a consequence, the only means to ensure a gratifying experience for a majority interacting users—especially for long-term relationships—would seem be the availability of a sufficiently rich sample of varied personalities, from which to identify and select fitting partners. Needless to say, that very selection process poses challenges of its own [9,48].

4 Conclusion: Our Own Concerns

Any survey cannot be complete without at least some cautioning observations and references to problems or dangers inherent in the domain at hand. The first observation was put forward by Microsoft's Gene Ball [6] but is easily shared by the authors: as with any novel research area that forms a current focus of attention, there is the immanent risk that this high level of interest may generate an unreasonably high level of expectation. In the present case, an expectation for technology that dramatically and reliably interprets emotional behaviour. More likely than not, true impact of emotionally aware computing will be in smoothening many of today's rough edges in interaction with information technology (see also [58]). Second, emotion can be a very powerful persuasive tool. This aspect becomes all the more relevant if connected to never-slackening technology one personally owns and one interacts with extensively (cf. e.g. [7]). A related research programme was recently initiated at Stanford University under the direction of B.J. Fogg [26] at Stanford University, under the name of Computer Aided Persuasive Technology (CAPTology).

But to conclude—with all due caution—on an optimistic note: in particular in combination with agent technology, affective computing holds the promise to be one of the most satisfyingly useful technologies coming about.

5 Acknowledgement

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry Education, Science and Culture. This work is funded partly from project GZ 61.096/4-V/B/99 of the Austrian Federal Ministry of Transport, Innovation and Technology, and from the SAFIRA project (IST-1999-11683) of the 5th Framework Programme of the European Union. The authors wish to acknowledge in particular the contributions provided by the original work of Alexander Staller.

References

1. Agre, P., Horswill, I.: Lifeworld Analysis. JAIR **6** (1997) 111–145
2. Arnold, M.B.: Emotion and Personality. Columbia University Press, New York (Vols. I and II) (1960)
3. Arzi-Gonczarowski, Z.: A Categorization of Autonomous Action Tendencies: The Mathematics of Emotions. In: [75] 683–688
4. Ball, G., Ling, D., Kurlander, D., et al.: Lifelike Computer Characters: The Persona Project at Microsoft Research. In: Bradshaw, J.M. (ed.): Software Agents. AAAI Press/The MIT Press, Menlo Park CA (1997) 191–222
5. Ball, G., Breese, J.: Emotion and Personality in a Conversational Character. In: Cassell et al. (eds.): Embodied Conversational Agents, MIT Press, Cambridge MA (1999) 189-219
6. Ball, G.: A Bayesian Heart: Computer Recognition and Simulation of Emotion. In: [76]

7. Bargh, J.A., Chartrand, T.L.: The Unbearable Automaticity of Being. American Psychologist **54***(7)* (1999) 462–479
8. Bordwell, D.: Narration in the Fiction Film. Univ. of Wisconsin Press, Madison WI (1985)
9. Boyce, S.J.: Natural Spoken Dialogue Systems for Telephony Applications. Comm. of the ACM **43***(9)* (2000) 29–34
10. Breazal, C.: Sociable Machines: Expressive Social Exchange Between Humans and Robots. Sc.D. Dissertation. MIT Dept. of EECS, Cambridge MA (2000)
11. Cañamero, D. (ed.): Emotional and Intelligent: The Tangled Knot of Cognition, Proc. of 1998 AAAI Fall Symp., Orlando FL, AAAI (1998)
12. Cañamero, D., Numaoka, C., Petta, P. (eds.): Grounding Emotions in Adaptive Systems. Workshop Notes, 5th Int'l Conf. of the Soc. for Adaptive Behaviour (SAB'98), Zurich Switzerland. Austrian Res. Inst. for Artif. Intell., Vienna Austria EU (1998) `http://www.oefai.at/oefai/agents/`
13. Cañamero, L.D., Petta, P. (eds.): Grounding Emotions in Adaptive Systems. (Vol I) Cybernetics and Systems **32***(5)* (2001)
14. Churchland, P.M., Churchland, P.S.: On the Contrary: Critical Essays, 1987-1997. MIT Press/Bradford Books, Cambridge MA London England (1998)
15. Clancey, W.J.: Situated Cognition: On Human Knowledge and Computer Representations. Cambridge University Press (1997)
16. Dalgleish, T., Power, M. (eds.): Handbook of Cognition and Emotion. Wiley, Chichester London New York, 1999.
17. Damásio, A.R.: Descartes' Error. Grosset/Putnam, New York (1994)
18. Damásio, A.R.: The Feeling of What Happens: Body and Emotion in the Making of Consciousness. Harcourt Brace Jovanovich, New York (1999)
19. Darwin, C.: The Expression of the Emotions in Man and Animals (1872) Republished Univ. of Chicago Press, Chicago IL (1965) `http://etext.lib.virginia.edu/ebooks/`
20. Ekman, P.: Basic emotions. In: [16] 45–60
21. Elliott, C.D.: The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System. Ph.D. Diss. Northwestern Univ., Evanston IL (1992)
22. Elliott, C.: Hunting for the Holy Grail with "Emotionally Intelligent" Virtual Actors. ACM Intelligence **1***(1)* (1997)
23. Elliott, C., Brzezinski, J.: Autonomous Agents as Synthetic Characters. AI Magazine, **19***(2)* (1998) 13–30
24. Elliott, C., Rickel, J., Lester, J.: Lifelike Pedagogical Agents and Affective Computing: An Exploratory Synthesis. In: [79] 195–212
25. Ferreira, C.P., Ventura, R., Petta, P.: Autonomous Control: Lessons from the Emotional (ACE-2000). Symposium Proceedings. In: [75]
26. Fogg, B.J.: Persuasive Technologies. Comm. of the ACM **42***(5)* (1999) 26–29
27. Foner, L.N.: What's An Agent, Anyway? MIT Media Laboratory, Cambridge MA (1993)
28. Frank, R.H.: Passions within Reason. W.W. Norton, New York (1988)
29. Frijda, N.H.: The Emotions. Cambridge University Press, Paris (1986)
30. Frijda, N.H., Mesquita, B., Sonnemans, J., van Goozen, S.: The Duration of Affective Phenomena or Emotions, Sentiments and Passions. In: Strongman, K.T. (ed.): Int'l Review of Studies on Emotion (Vol. 1). Wiley, Chichester (1991) 187–225
31. Frijda, N.H. (ed.): Appraisal and Beyond: The Issue of Cognitive Determinants of Emotion. Cognition & Emotion **7***(3&4)* (1993)
32. Frijda, N.H.: The Place of Appraisal in Emotion. Cognition & Emotion **(7)***(3&4)* (1993) 357–388

33. Frijda, N.H.: Emotions in Robots. In: Roitblat, H.L., Meyer, J.-A. (eds.): Comparative Approaches to Cognitive Science. MIT Press/Bradford Books, Cambridge London (1995) 501–516
34. Gross, J.J.: Emotion Regulation: Past, Present, Future. Cognition & Emotion **13***(5)* (1999) 551–573
35. Halperin, J.R.P.: Cognition and Emotion in Animals and Machines. In: Roitblat, H., Meyer, J.-A. (eds.): Comparative Approaches to Cognitive Science. MIT Press, Cambrigde MA (1995)
36. Hendrix, J., Rukktay, Zs., ten Hagen, P., Noot, H.,Lelievre, A., de Ruiter, B.: A facial repertoire for avatars. Proc. Workshop Interacting Agents, Enschede The Netherlands, 2000
37. Jensen, F.V.: An Introduction to Bayesian Networks. Springer-Verlag, New York (1996).
38. Keltner, D., Haidt, J.: Social Functions of Emotions at Four Levels of Analysis. Cognition & Emotion **13***(5)* (1999) 505–521
39. Kemper, T.D.: Sociological Models in the Explanation of Emotions. In: [43] 41–52
40. Lazarus, R.S.: The cognition-emotion debate: a bit of history. In: [16] 3–19
41. LeDoux, J.E.: The Emotional Brain. Simon & Schuster, New York (1996)
42. Lewis, M.: The Emergence of Human Emotions. In: [43] 223–237
43. Lewis, M., Haviland, J.M. (eds.): Handbook of Emotions. Guilford Press, New York London (1993)
44. Loyall, A.B.: Believable Agents: Building Interactive Personalities. Ph.D. Thesis. CMU School of Computer Science (1997)
45. Martinho, C., Paiva, A.: "Underwater Love": Building Tristão and Isoldas Personalities. In: [79] 269–296
46. Minsky, M.: The Society of Mind. Simon & Schuster (1988)
47. Moffat, D.: Personality Parameters and Programs. In: [74] 120–165
48. Nass, C., Gong, Li: Speech Interfaces from an Evolutionary Perspective. Comm. of the ACM 43, 9 (Sep. 2000), Pages 36 - 43.
49. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press Cambridge UK (1988)
50. Ortony, A., Turner, T.: What's basic about basic emotions? Psychological Review **97** (1990) 315–331
51. Petta, P.: Synthetic Characters and Affective Computing. Invited Presentation at SimHAF, (First) AgentLink/i3 Special Interest Meeting on Human/Agent Factors, 29.6.1999, Valencia, Spain. Austrian Res. Inst. for Artif. Intell. (1999) `http://www.ai.univie.ac.at/oefai/agents/simhaf/`
52. Petta, P.: Personality and Believability. Panel session at VA99, 2nd Workshop on Int. Virtual Agents. The Centre of Virtual Environments, Univ. of Salford Salford UK, 13th September, 1999.
`http://www.ai.univie.ac.at/oefai/agents/va99panel/`
53. Petta, P.: Principled Generation of Expressive Behavior in an Interactive Exhibit. In: [77] 94–98
54. Petta, P., Cañamero, L.D. (eds): Grounding Emotions in Adaptive Systems. (Vol II) Cybernetics and Systems **32***(6)* (2001)
55. Pfeifer, R.: The Fungus Eater Approach to Emotion: a View from Artificial Intelligence. Cognitive Studies **1** (1994) 42–57
56. Pfeifer, R.: Building "Fungus Eaters": Design Principles of Autonomous Agents. In: Maes, P., et al. (eds.): From Animals to Animats 4. MIT Press/Bradford Books, Cambridge London (1996) 3–12

57. Picard, R.W.: Affective Computing. MIT Press, Cambridge Boston London (1997)
58. Picard, R.W.: What Does It Mean for a Computer to "Have" Emotions? In: [76]
59. Reeves, B., Nass, C.: The Media Equation: How People Treat Computers, Television, and New media Like Real People and Places. Cambridge Univ. Press (1996)
60. Reilly, W.S.N.: Believable Social and Emotional Agents. Ph.D. Thesis. CMU School of Computer Science (1996)
61. Rizzo, P.: Emotional Agents for User Entertainment: Discussing the Underlying Assumptions. In: Int'l Workshop on Affect in Interactions: Towards a New Generation of Interfaces, Siena Italy (1999)
62. Rolls, E.T.: Brain and the Emotion. Oxford University Press, London Oxford New York (1999)
63. Roseman, I.J., Antoniou, A.A., Jose, P.E.: Appraisal Determinants of Emotions: Constructing a More Accurate and Comprehensive Theory. Cognition & Emotion **10***(3)* (1996) 241–277
64. Scherer, K.R.: Criteria for emotion-antecedent appraisal: A review. In: Hamilton, V., Bower, G.H., Frijda, N.H. (eds.): Cognitive Perspectives on Emotion and Motivation. Kluwer, Dordrecht (1988) 89–126
65. Scherer, K.R.: Appraisal Theory. In: [16] (1999) 637–663
66. Scherer, K.R., Schorr, A., Johnstone, T. (eds.): Appraisal Processes in Emotion: Theory, Methods, Research. Oxford Univ. Press (1999)
67. Sengers, P.: Anti-Boxology: Agent Design in Cultural Context. Ph.D. Thesis. CMU School of Computer Science (1998)
68. Simon, H.A.: Motivational and emotional controls of cognition. Psychological Review **74** (1967) 29–39
69. Sloman, A.: Motives, Mechanisms and Emotions. Cognition & Emotion **1** (1987) 217–234
70. Sloman, A.: Architectural Requirements for Human-like Agents both Natural and Artificial (What Sorts of Machines Can Love?). In: Dautenhahn, K. (ed.): Human Cognition and Social Agent Technology. John Benjamins Publishing (1999)
71. de Sousa, R.: The Rationality of Emotion. MIT Press, Cambridge Boston London (1987)
72. de Sousa, R.: Prefrontal Kantians. A Review of "Decartes' Error: Emotion, Reason and the Human Brain" by Antonio R. Damásio. Cognition & Emotion **10***(3)* (1996) 329–333
73. Staller, A., Petta, P.: Introducing Emotions into the Computational Study of Social Norms: A First Evaluation. Journal of Artificial Societies and Social Simulation, **4***(1)* (2001)
74. Trappl, R., Petta, P. (eds.): Creating Personalities for Synthetic Actors. LNAI 1995. Springer-Verlag, Berlin Heidelberg New York (1997)
75. Trappl, R. (ed.): Cybernetics and Systems 2000: Proc. of the 15th European Meeting on Cybernetics and Systems Research (EMCSR-2000). Austrian Society for Cybernetics Studies, Vienna Austria (2000)
76. Trappl, R., Petta, P. (eds.): Emotions in Humans and Artifacts. MIT Press, Cambridge MA (forthcoming)
77. Velasquez, J.D. (ed.): "Emotion-Based Agent Architectures" (EBAA'99). Workshop Notes, 3rd Int'l Conf. on Autonomous Agents (Agents '99), Seattle, WA, MIT Artifial Intelligence Lab (1999)
78. Weizenbaum, J.: ELIZA—A Computer Program for the Study of Natural Language Communication Between Man and Machine. Comm. of the ACM **9***(1)* (1966) 36–45
79. Wooldridge, M., Veloso, M. (eds.): Artificial Intelligence Today: Recent Trends and Developments. LNAI 1600. Springer-Verlag, Berlin Heidelberg New York (1999)

Multi-agent Coordination and Control Using Stigmergy Applied to Manufacturing Control

Paul Valckenaers, Hendrik Van Brussel, Martin Kollingbaum, and Olaf Bochmann

K.U.Leuven – P.M.A., Celestijnenlaan 300B, B-3001 Leuven, Belgium
{Paul.Valckenaers, Hendrik.VanBrussel, Martin.Kollingbaum, Olaf.Bochmann}@mech.kuleuven.ac.be

Abstract. This manuscript discusses multi-agent coordination and control using techniques inspired by the behavior of social insects. It presents a system design that enables desirable overall behavior to emerge without exposing the individual agents to the complexity and dynamics of the overall system. The research, which this paper discusses, focuses on manufacturing control. However, the approach remains applicable to the coordination and control of other types of ironware systems (e.g. traffic, supply chain…).

1. Introduction

Manufacturing control handles the internal logistics in a manufacturing system. It decides about all the routings of product instances as well as the starting of production processes on these unfinished products. Manufacturing control software developers face challenges posed by the changes and disturbances in the manufacturing system and its environment. Moreover, this software lacks the number of users that mainstream software enjoys. Therefore, lower investment levels and less user feedback represent additional barriers for the successful development and deployment of this type of software.

This paper discusses a multi-agent coordination and control system design, inspired by the behavior of social insects, which aims to answer these challenges. In particular, it makes desirable overall system behavior emerge without exposing the individual agents to the complexity and the dynamics of the overall system. This enables the individual agents to survive changes without software maintenance (= live longer), it allows individual agents to be re-usable across systems, and it allows having the emergent behavior handle disturbances.

The paper starts with the biological concept, stigmergy, that constitutes the basis of the coordination and control system. Next, it discusses the different steps in the development of a coordination and control system:

1. Agentify the environment to make it part of the solution.
2. Implement the control layers that make necessary and useful information available to the agents that take the decisions.
3. Implement the decision taking mechanisms at the highest level in the control system, such that changing and adapting these rules becomes very easy.

2. Stigmergy

P.P. Grassé introduced the word stigmergy in 1959 [1][2]. Stigmergy means that agents put signs, called stigma in Greek, in their environment to mutually influence each other's behavior. Such mechanism is suitable for small-grained interactions compared to coordination methods that require an explicit rendezvous amongst the agents. With stigmergy, agents observe signs in their environment and act upon them without needing any synchronization with other agents.

Coordination between agents occurs when they adapt their activities because of interactions [3]. These interactions can be direct or indirect. Stigmergy belongs to the category of indirect interactions. The situation is analogous to one person buying a kilo of apples in a supermarket. The display of the apples and their price constitute the signs in the environment. The agent observes these signs and decides whether he will buy these apples without direct interaction with any other agent.

Note also that the signs in the environment typically are multi-dimensional and reflect relevant aspects of the coordination task at hand; the display of apples provides the agents with information through the look, smell and packaging of the apples. In contrast, many market-based approaches reduce the dissipative field [4] to a single dimension (= money) [5][6].

This paper addresses a specific kind of stigmergy: the signs that are locally available in the environment allow agents to learn about global properties of the system. Importantly, these signs are put in the environment without exposing individual agents to the complexity and the dynamics of the situation. The biological source of inspiration that reveals how this can be done consists of food foraging ants. This source of inspiration already gave cause to a collection of research results [7-12]. The following paragraph introduces this type of stigmergy and discusses its particular achievement.

2.1 Food Foraging Ants

Food foraging ants execute a simple procedure in which their behavior is guided by a permanently changing environment. Ants forage for food in the following way:

- In absence of any signs in the environment (consisting of by scents from a chemical substance called a pheromone), ants perform a randomized search for food.
- When an ant discovers a food source, it drops a chemical smelling substance—i.e. pheromone—on its way back to the nest while carrying some of the food. Thus it creates a pheromone trail between nest and food source. An important property of such pheromone trail is that it will eventually evaporate if no other ant deposes fresh pheromones.
- When an ant senses signs in form of a pheromone trail it will be urged by its instinct to follow this trail to the food source. The ant’s behavior always remains probabilistic: there is a high probability that it follows the found trail, but no certainty, where the probability depends on the strength of the pheromones. When the ant finds the food source, it will return with food, while deposing pheromones itself. The strength of the pheromone trail is maintained and even reinforced. When

the ant discovers that the food source is exhausted, it starts a randomized search for food and the trail disappears because of the evaporation.

These simple behavior patterns result in an emergent behavior of the ant colony that is highly ordered and very effective at foraging food while being robust against the uncertainty and complexity of the environment.

An important capability of this type of stigmergy is illustrated here: global information (about where to find food in a remote location) is made available locally (in which direction must the ant move to get to this food). At any location in the environment of the ant where pheromone trails exist, the ant learns about the availability of food in remote locations.

The main achievement is that individual ants are not exposed to the complexity and dynamics of this situation. Indeed, the environment itself is incorporated into the solution and allows the overall system to cope with the complexity of the environment: the complexity of the pheromone trails is handled by putting them into the environment itself. None of the ants needs a mental map of the environment nor do these ants communicate amongst each other about the environment. Similarly, the evaporation and refreshing of the pheromone trails allows the ants to cope with the dynamics of the environment (e.g. a farmer modifying the surface geometry). Moreover, there are no mental maps in the head of the ants that need to be kept in sync with reality.

The remainder of this paper discusses how the above advantages can be achieved in multi-agent coordination and control, especially for manufacturing control. Nonetheless, the presented mechanisms remain applicable to a wider range of situations: the coordination and control of ironware systems. Ironware means that the underlying system is operating much slower than the computer system on which the multi-agent coordination and control system executes. Moreover, the ironware system does not compete for computer resources with the agent system, and the ironware system is subject to physical laws and technological constraints. A detailed discussion of the applicability range of the multi-agent coordination and control techniques, which are presented here, is however outside the scope of this paper.

3. Environment as a Part of the Solution

The first step in the development of an ant-based solution is to make the environment a part of the solution. The developments, on which this manuscript reports, apply a specific architecture—PROSA—dividing the agents in four different categories. Detailed information can be found in [13][14]. The next section gives an overview of this architecture. Next, the discussion introduces resource agents that make the environment a part of the solution.

3.1 The PROSA Reference Architecture

PROSA is a reference architecture; it provides a starting point for the design and development of multi-agent manufacturing control systems. The structure of the architecture is built around three types of basic agents: order agents, product agents, and resource agents. Each of them is responsible, respectively, for one aspect of manufacturing control: (i) internal logistics, (ii) recipes or process plans, and (iii) resource handling. These basic agents are structured using object-oriented concepts like aggregation and specialization. Staff agents can be added to assist the basic agents with expert knowledge. The reference architecture is called PROSA, which stands for Product-Resource-Order-Staff Architecture and refers to the composing types of agents.

Each resource agent corresponds to a physical part—a production resource in the manufacturing system—and contains an information processing part that controls the resource. It offers production capacity and functionality to the surrounding agents. It holds the methods to allocate the production resource, and the knowledge and procedures to organize, use and control the production resource to drive production. A resource agent is an abstraction of the production means such as a factory, a shop, machines, furnaces, conveyors, pipelines, pallets, components, raw materials, tools, tool holders, material storage, personnel, energy, floor space, etc. In object-oriented terminology, each resource agent reflects a physical resource, is able to drive this resource, and keeps itself in sync with this resource.

Each product agent holds the process and product knowledge to assure the correct making of the product with sufficient quality. A product agent contains consistent and up-to-date information on the product life cycle, user requirements, design, process plans, bill of materials, quality assurance procedures, etc. As such, it contains the "product model" of the product type, not the "product state model" of one physical product instance being produced. The product agent acts as an information server to the other agents, delivering the right recipes in the right place. It knows how to make a product without making any decisions about when and where production takes place.

Each order agent represents a task in the manufacturing system. It is responsible for performing the assigned work correctly and on time. It manages the physical product being produced, the product state model, and all logistic information processing related to the job. An order agent may represent customer orders, make-to-stock orders, prototype-making orders, orders to maintain and repair resources, etc. Often, the order agent can be regarded as the work piece with a certain control behavior to manage it to go through the factory, e.g. to negotiate with other parts and resources to get itself produced.

Interaction. These three types of agents exchange knowledge about the manufacturing system. Product agents and resource agents communicate process knowledge, product agents and order agents communicate production knowledge, and resource agents and order agents share process execution knowledge.

Aggregated agents are defined as a set of related agents that are clustered together and that form a bigger agent with its own identity. The aggregation hierarchy seldom is tree-shaped; agents may belong to multiple aggregations (e.g. a tool can be shared between several workstations). Aggregated agents are no static sets of agents, but can dynamically change their contents depending on needs of the system. For instance, membership of a batch order agent may depend on the timely arrival of the prospective members. Aggregated agents may emerge out of the self-organizing interaction of agents or they may be designed up front. The number of hierarchical levels depends on the specific needs of a certain system, and is not dictated by the architecture.

Specialization classifies the agents by their characteristics. The architecture defines a top-level specialization by separating the basic agents into three categories: order agents, product agents and resource agents. In a specific architecture, these basic agents may still be too abstract to reason about. Specialization can then be used to differentiate between the different kinds of resource agents, of order agents, and of product agents. For instance, shop, workstation, and equipment are all resource agents. Similarly, specialization can be used to structure the order and product types. Order agents can represent customer orders, stock orders or maintenance tasks. Products can be divided into product families.

Staff agents. Peppard [15] identifies three types of processes within business organizations: strategic, operational and enabling processes. The basic agents constitute the operational processes in the manufacturing system. The strategic processes are outside the scope of the multi-agent coordination and control developments. There remains, however, a need for the enabling processes; staff agents will provide these. The name "staff agent" is inspired by the difference between line functions and staff functions in human organizations. In a human organization one of the main goals for the introduction of staff functions is to reduce the workload and complexity of line functions (or operational processes) by providing them with expert knowledge.

Accordingly, the PROSA architecture allows for staff agents to assist the basic agents in performing their work. They provide the basic agents with information such that they can take better decisions. The basic agents are responsible for taking the decisions; the staff agents are external experts giving advice. Note that this manner of cooperation avoids the rigidity of conventional designs. When a staff agent is unable to perform its task adequately, or is unable to respond fast enough, the basic agents autonomously will take a default decision and/or other appropriate actions (probably sub-optimal but safe). The concept of staff agents allows for the presence of specialist elements and functionality in the architecture. It also permits to achieve advantages of scope, in which one staff agent implementation serves in many systems where an implementation as a basic agent would require extensive customization at every installation.

3.2 Resource Agents

In an ant-based system, the environment is part of the solution. In a telecommunications application, the real computer communication network can be used for this purpose. This is possible because it is a part of the universe in which the agents live. However, in manufacturing, the real system is ironware and this system needs to be connected to the world in which the agents reside.

Therefore, the first step in the development of a multi-agent manufacturing control system consists of mirroring the manufacturing system in the agents' world. For every resource or entity in the factory, a resource agent is created. This agent is knowledgeable about its corresponding resource and keeps itself in sync with reality—i.e. the resource agent observes and tracks the state of the corresponding resource. In object-oriented terminology, the factory resources are reflected into resource agents.

In addition, the resource agents offer to the other agents some data storage spaces on which these agents can put, modify and retrieve information (=pheromones). Such information will have a time-bounded lifespan (=evaporation).

Moreover, it is possible to take advantage of the fact that the agents are much faster than the underlying ironware, which is not competing for the computer resources with the agents. Resource agents will offer support for what-if analysis: they mirror the behavior of their physical resource in a virtual world. This is elaborated below, especially in the section on emergent load forecasting.

Importantly, the exposure of such resource agents is limited to the corresponding resource. Wherever such a resource resides, the agent's software code can be reused. It is exactly this property that makes the ant-based control system designs attractive and worthwhile. Note that resource agents can become sophisticated experts, but with a limited scope: the resource. The following illustrates this development step with the discussion of a sample resource agent.

A Conveyor Belt Agent reflects an accumulating conveyor belt on which products are transported in a single direction and are unable to overtake each other. It is a first-in, first-out or FIFO device with a finite capacity. The following paragraphs discuss the functionality supported by this conveyor belt agent.

Attributes. The conveyor belt agent provides a collection of local blackboards. These blackboards correspond to the physical places on which the ants put the pheromones. Agents write, read and remove information from these blackboards. The infrastructure supports evaporation over time for information on the blackboards. Each blackboard can, for instance, be implemented by means of a tupple space. In addition, the resource agent has attributes to model its state. The sample conveyor agent provides the following attributes:

- Blackboard connected to the belt entrance
- Blackboard connected to the belt exit
- Blackboard connected to the middle section
- State vector including references to the entities on the belt
- Graph supporting (virtual) navigation across the conveyor including references to the entities connected to respectively the conveyor entry and exit
- …

Life Cycle Support. It is important to cover the full life cycle if the manufacturing control system has to cope with changes; with incomplete support the system will have time windows in which it will be extremely vulnerable. Access to these life cycle support functions is subject to proper authorization The sample conveyor belt agent supports the following:

- Installation in the factory (creation)
- Removal from the factory (destruction)
- Connecting the entrance to a resource
- Connecting the exit to a resource
- Disconnecting the entrance
- Disconnecting the exit point
- Updating the state vector to synchronize with reality
- Setting belt speed
- …

State observers. As in the design of any class of objects, resource agents support methods/functions to observe their state. As a rule, it is preferable to offer observation methods over direct access to an attribute; the method can control access to the attributes more effectively and it encapsulates the implementation, which may need to change later. Sample observation methods offered by the sample conveyor belt agent are:

- Query the current load
- Query belt speed
- …

Event processing and notification. An agent is an active entity, capable of triggering actions. For instance, the sample conveyor agent will notify the corresponding order agent when a product instance reaches the conveyor exit. Conversely, the conveyor agent will process the arrival event of such a product at its entry, possibly informing the corresponding order agent that the conveyor is full and that the product is unable to enter. In this situation, the order agent probably requests to be notified when space becomes available on the conveyor.

What-If Analysis. The resource agent has a moderate level of intelligence, in contrast to the passive environment of the ants. The resource agent provides reflection functionality—self-knowledge and self-modeling—that enables other agents to cooperate with this agent during what-if analysis:

- `PropagateUpstream` will move an (mobile) agent from the exit to the entrance and adjust time information, which is presented to this conveyor agent, to reflect the expected transportation delay on the belt; time data will indicate earlier times at the entrance. The agent is then directed to the resource connected to the entrance (if any).
- `PropagateDownstream` will move an agent from the entrance to the exit and adjust time information, which is presented to the conveyor agent, to reflect the expected transportation delay on the belt; time data will indicate later times at the exit. The agent is then directed to the resource connected to the exit (if any).
- `GiveLoadForecast` will generate a (short-term) forecast of the resource usage or loading based on intentions that have been communicated by prospective users; these users communicate their intentions by means of the two previous functions, indicating their level of commitment through the parameter settings.

In the current developments, information about time consists of a single numerical value. In future versions, the agents can become more intelligent and able to handle probabilistic information as well; this would enable the system to represent the confidence level of the forecasts. Note that a designer is not forced to choose what information is propagated. A system can propagate several kinds of information while order agents dynamically decide which kind they use, or, with simple agents ignoring the complex information while more capable agents use all available data.

Discussion. Note how this resource agent remains fully functional as long as he reflects the corresponding physical resource and as long as his internal representation of natural laws (time, space, probability…) stays valid. Beyond that, the agent has minimal exposure—with the exception of superficial matters (e.g. syntax issues) for which technical solutions exist. Consequently, resource agents are likely to be long-lived, capable of evolving with the corresponding resource, maximize their number of possible users, capable of becoming more intelligent in their own domain, and capable of surviving with little maintenance.

Other resource agents mirror switching elements in the transport system, storage and retrieval subsystems, processing units… Together, these agents provide an infrastructure on which the order agents place and observe information to coordinate their activities and optimize overall behavior. The next sections discuss how the order agents use this infrastructure.

4. Control Layers

The multi-agent manufacturing control developments, on which this paper reports, have a layered design. In each layer, information is made available and/or exchanged to enable the order agents to make proper choices regarding the internal logistics of the manufacturing system: which route to choose, which processing step to start at which processing unit, whether and when to join a batch… The order agents combine the information provided by the layers to make their decisions. In principle, the information, which the layers provide, consists of facts rather than decisions; note however that intentions of agents are considered to be facts (see below).

Each individual implementation consists of a particular combination of control layers. The individual control layers are designed to be re-usable; the re-usability of a particular combination will be much lower. Because of this, the order agent implementations don't care about the presence of control layers that they do not use. As a consequence, a control system will function while some obsolete control layers are present and active in the system. This will squander computer resources but reduces the chance that a layer, e.g. needed in rare situations, is removed inadvertently. And, a currently unused control layer, which makes relevant facts available, allows the rapid introduction of new decision mechanisms that utilize this information about these facts. Furthermore, future implementations may contain order agents that adapt to the availability and quality of information from the control layers.

The control layers belong to three different categories. The first type of layers addresses feasibility. These layers deliver information that cannot be ignored because it reflects hard constraints. The second type of layers provides information to operate the system smoothly and with good performance. These layers provide, among others, information about the availability/capacity of the resources. This information is needed to operate the system well. The third type of layers corresponds to the staff agents in the PROSA architecture. These layers typically provide advisory information to guide the search process of the order agents but without carrying the final responsibility. These categories are discussed below.

4.1 Feasibility Layers

The primary concern of order agents is to get their product instance(s) manufactured. To achieve this goal, the hard constraints in the manufacturing system have to be accounted for. In research developments, two major categories of hard constraints were encountered: recipes (or process plans) and deadlock situations. The former is always present whereas the presence of the latter depends on the manufacturing system.

Recipe Layer. All manufacturing control systems must enforce the appropriate recipes (or process plans). To this end, a control layer provides the necessary information on the local blackboards of the resources wherever order agents have to make routing decisions. This information enables the order agent to know which routing options are feasible with respect to the product recipe.

This information depends on the type of factory. In factories with a flexible transport system, it suffices to check whether a routing option leads to (one of) the (possible) next processing step(s). In more rigid systems, it typically is necessary to check also whether all subsequent processing steps remain reachable when a particular routing option is selected [11]. In general, the particular layer that is required will be system-dependent, but many layers will be re-usable across manufacturing systems.

Consider a manufacturing system in which it suffices to know whether the next processing step is reachable in a particular direction. In this case, a simple ant-like information spreading mechanism will perform the task adequately:

- The resource agents of processing units create, at a certain frequency, a mobile agent and send it upstream starting at the entry of their processing resource. These mobile agents put information on the local blackboards describing the processing capabilities of the corresponding processing unit. This enables order agents to distinguish which processing capabilities are available along a given route.
- The mobile agents clone themselves when a transportation resource has multiple entries.
- The mobile agents are able to detect loops and stop propagating as soon as they detect that they already visited a location.
- When a mobile agent arrives at another processing unit, resource reflection by the resource agent informs the mobile agent whether the processing unit supports a *no-operation*. If the processing unit fails to support this, upstream propagation ends; otherwise, propagation continues reflecting the fact that the processing unit can function as a transport device.

Note that the evaporation mechanism will remove outdated information. Resource agents corresponding to processing units have to refresh the information regularly. Of course, these resource agents can also wipe out information when it is no longer valid, but this is insufficient reason to discard the evaporation and refresh mechanism. Indeed, if some upstream part of the manufacturing system becomes unreachable, stale information is not removed. The evaporation and refresh guarantees robustness. Explicit removal of outdated information must only be used to decrease the delay with which information becomes available.

Deadlocks. A second concern for the order agents is to keep the system out of deadlock states. To prevent or avoid deadlocks, order agents have to exchange information about their ownership of resources as well as their intentions to acquire additional resources while holding these resources. Order agents have to cooperate, following a specific mechanism adapted to their situation in this matter; indeed, known generic solutions in this domain are NP hard and not practical.

In practice, many manufacturing systems are organized such that deadlock cannot occur. The products flow through the production system in a single direction and resources are acquired in a single sequence. In such systems, there is no risk of circular hold-and-wait, which characterizes deadlock. In other systems, resources can be numbered while order agents know that they can only hold resources with low numbers while waiting for resources with high numbers. If order agents need to acquire these resources out of order, they still obtain the ownership rights in the prescribed sequence, possibly making resources idling unnecessarily but unavoidably. An example is given in [13]. Unfortunately, such simple solutions cannot be applied in all cases. A common exception is a transport system containing loops; all manufacturing systems in which products may visit the same processing units more than once, this non-trivial situation presents itself. When such loops fill themselves completely, the system will be deadlocked. An example of a mechanism to handle such situations is given in [16].

These deadlock-handling methods typically do not apply the food foraging ant mechanism in full. They still apply the incorporation of the (agentified) environment as part of the solution. For instance, the method described in [16] employs the resource agents to identify which agents are involved in a potential deadlock situation. However, there is no need to make global information available on local blackboards.

Furthermore, because generic solutions are NP hard, deadlock handling control layers will be system-specific. However, such control layers will be re-usable across manufacturing systems. In addition, such system-specific solutions are likely to make some implicit resource allocation choices; they will not always be maximally permissive. This means that they may violate the principle that layers only make information about facts available; good designs will avoid this as much as possible.

Finally, the re-usability of the deadlock handling layers has to be part of the order agents' behavior. These layers cannot be implemented as information that is provided to the order agents on local blackboards. Therefore, re-use of existing implementations will probably occur through inheritance of the behavior of deadlock handling agents by specific order agents.

4.2 Operation Layers

In an economic context, it is necessary but insufficient the get the right product instance made. It is also imperative to produce those instances fast and cheaply. A manufacturing control system must maximize throughput, limit lead-time, limit work-in-process, and respect priorities. Especially throughput, measured in financial income from the product instances that are delivered, is a critical. The operation layers provide the system with the proper information to achieve this. Note that these layers are concerned with making the actual decisions in the actual situation (= line function).

In the basic ant-based design [10], each order agent creates mobile agents and sends them upstream through the system, along the path which the order followed, where this mobile agent writes information on the local blackboards about the (lack of) success that the order agent had while having its product instance produced [7]. If the order gets stuck in queues and traffic jams, subsequent orders are discouraged from repeating its choices. If the order experiences little problems, subsequent orders are encouraged to follow its track. These design have the advantage of being simple and of requiring little communication overhead. They have been successfully applied for packet switching networks in telecommunications [12][8].

In manufacturing control, the implementation of the basic ant design presents some additional challenges. For instance, manufacturing systems make different types of products (shapes, colors, options) that follow different routes through the system. Therefore, order agents must decide which information is relevant for them.

Moreover, manufacturing control must face additional concerns. Examples are sequence-dependent set-ups, sequence-dependent yields, and product-dependent processing times. To answer such concerns, manufacturing control must be able to group products in the appropriate batches. In addition, large products cannot be stored nor re-sequenced arbitrarily. Carefully planned maneuvering is required. Extra control layers are needed to identify the opportunities for grouping product instances and for executing the maneuvers that will bring these instances in the proper sequence at the manufacturing resources. In other words, manufacturing control is much more sophisticated than the management of telecommunication networks (although this may change when quality-of-service is addressed in such systems).

The most important disadvantage of the basic ant colony design is that it does not anticipate. Indeed, the feedback from the mobile agents may arrive too late to be of any use; the situation may change too rapidly. Fortunately, the agents have the advantage that they are much faster than the ironware under their control. Section 5 illustrates how this can be used to forecast the effects of the agents' intentions on the load of the processing units to give the agents the opportunity to optimize their decisions. Similarly, forecasts can be created to identify opportunities to address the other concerns (e.g. to form or join a suitable batch when traveling toward a batch processing unit).

The ability to anticipate in agent-based coordination and control is typical for the control of ironware systems: agents are faster and, therefore, can emulated the system's behavior several times before the actual decision is taken. Moreover underlying the ironware does not compete for the computer resources while small gains in the critical performance criteria of the ironware system typically will pay for a sizeable computer infrastructure.

4.3 Staff Layers

Since they bear the final responsibility, the operation layers deal with information at a detailed level. If an order has some special requirements, the feasibility and operational levels must be able to handle this. The downside of this approach is that these layers do not see the bigger picture. To handle this drawback, the PROSA architecture provides staff agents. In a multi-agent manufacturing control system, these staff agents mainly operate in two places. First, they can influence the decision taking process of the order agent directly (section 6). Second, they can distribute information on the blackboards in the system. It is the latter that constitutes staff layers.

An analogy in the ants world would be flying ants that recognize spots with indications that they are food sources and that drop pheromones between such spots and the ant nest while flying in straight line through the air. Such flying ants are likely to speed up the exploration process of the food foraging ants on the ground, while evaporation eliminates all mistakes from the flying ants. The flying ants only need to be more or less right in most cases; they do not carry a final responsibility.

In the manufacturing control system, staff agents can play a similar role. They can for instance analyze the product mix versus the processing capacities of the resources, statically, and initiate a distribution of the different product types across the resources. This will help the operation layers and the order agents to achieve an organized behavior faster. However, the operation layer and order agents will correct any mistakes and negligence of the staff agent.

Experience shows that staff agents are added last to the control system. Experimental results on the effectiveness of staff agents are not expected to become available until research on the operation layers has achieved sufficient maturity.

5. Intention-Based Forecasting

This section addresses the particular advantage of coordinating and controlling ironware. Agents can emulate/forecast the behavior of the system by combining the intentions of the order agents with the self-modeling abilities of the resource agents. This section illustrates this capability for one particular aspect: load forecasting for the processing units where these forecasts are made available on the local blackboards suitably transformed for local decision taking.

5.1 Propagation of Intentions

Figure 1 shows how an order agent propagates his intentions downstream. First, the order agent, who stays together with the physical semi-finished product, creates a mobile agent who will represent the order in a what-if mode. This mobile agent—the ant in figure 1—will move ahead through the production system in a virtual manner, and will make the order agent's intentions known wherever it visits a resource.

The mobile agent starts moving downstream in a virtual manner. The agent uses the `PropagateDownstream` function of the resources on which he travels. This enables the mobile agent to predict arrival times with minimal knowledge about the system through which he travels (minimal exposure). Through parameter settings, when calling `PropagateDownstream`, the mobile agent informs the resource agent about the commitment level for the order agent's intentions.

Whenever the mobile agent reaches a decision point, it executes the order agent's decision mechanism. During this execution, the mobile agent uses the forecast information at a time corresponding to his expected arrival. This expected arrival time is calculated by means of calling `PropagateDownstream` while moving through the system. Recall that the mechanism presented in this section constructs short-term forecasts and makes them available locally. Therefore, it is safe to assume that these forecasts are available on the local blackboards of the resource agents.

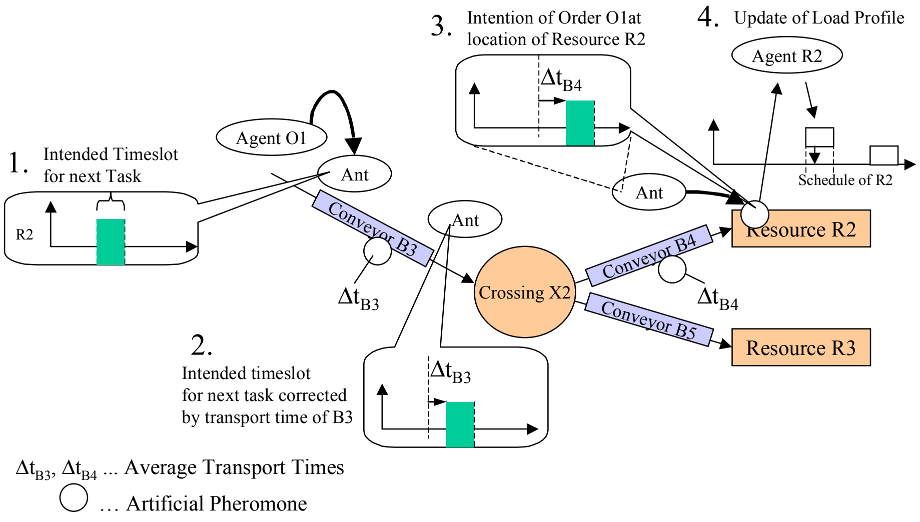

3. Intention of Order O1at location of Resource R2
4. Update of Load Profile
Agent R2
Δt_{B4}
Agent O1
1. Intended Timeslot for next Task
Ant
R2
Conveyor B3
Ant
Ant
Schedule of R2
Resource R2
Conveyor B4
Crossing X2
Δt_{B3}
Δt_{B4}
Conveyor B5
Resource R3
2. Intended timeslot for next task corrected by transport time of B3
Δt_{B3}
Δt_{B3}, Δt_{B4} ... Average Transport Times
… Artificial Pheromone

Fig. 1. Propagation of intentions.

Note the encapsulation of the order agent's decision mechanism. If it changes, the propagation of intentions is automatically adapted to the new situation; no software maintenance is necessary for the mobile agent. Also note that the system evaporates information that is not refreshed regularly. In this situation, evaporation is binary: after a timeout, it is discarded completely. This means that an agent can change his intentions without worrying about leaving stale data about previous intentions around. In practice, the mobile agent may explicitly wipe out stale data to make the new situation known faster; however, it suffices to use a wiping mechanism that covers most of the ground most of the time.

Eventually, the mobile agent reaches a (critical) processing unit and makes the order agent's intentions known. Figure 1 shows how these intentions change when they move through the system. The order agent wants to start the next processing step as soon as possible (i.e. now). When the mobile agent propagates this, the expected transportation times are added. At arrival on the processing unit, the corresponding resource agent knows when processing can start.

In addition, the mobile agent establishes a link between this resource agent and its product agent such that the expected processing time can be calculated. This information allows the resource agent to update the local load profile forecast (step 4 in figure 1). Indeed, by combining the intentions, which mobile agents communicate to the resource agent, the local short-term load forecast for the processing unit can be constructed. This mechanism is able to avoid exposure of individual agents to the complexity and the dynamics of the overall system.

5.2 Back-Propagation of Load Forecasts

Similar to the order agents, resource agents create mobile agents. Normally, this function is only activated for the more critical resources (mostly processing units). These mobile agents propagate the local load forecasts upstream through the system. The mobile agents, created by resource agents, leave the corresponding resource through the entrance (in a virtual manner) and use the `PropagateUpstream` method from resource agents to travel backwards through the factory. The mobile agents take the local load forecast for their resource with them and deposit copies of it on the appropriate blackboards; blackboards that correspond to a routing choice definitely receive a copy of the forecast from the mobile agent. However, these local load forecasts are adapted during the upstream propagation.

The local load forecasts have their time information adapted when they are propagated upstream. Figure 2 shows how time information is made earlier in the load forecast by the amount Δt_{B4}. When the local load forecast is propagated upstream, the adapted load profile shows an occupation/availability forecast, for the down-stream resource, for times at the location of the blackboard taking into account the remaining transportation time to the resource.

This changes a local load forecast into an opportunity forecast at the location of the blackboard on which the information is posted. An order agent at the entrance of conveyor B4 only has to look at the position with time equal to zero (= now), in this opportunity forecast, to see what the expected load is when he will arrive at the resource. Mobile agents created by order agents use their expected arrival time at the entrance of B4.

At entrance of a conveyor, this information is not very useful. However, when an agent arrives at the entrance of Crossing X2 (figure 2), the agent can look at the blackboards at the exits of X2 and compare opportunities. The routing decision across X2 can depend on these forecasts.

Importantly, the order agents and their mobile agents do not need to know the route toward a resource when they make their routing decisions. The load forecasts on the local blackboards are sufficient. Similarly, the resource agents and their mobile agents do not need to know about routing through the system, because they use the `Propa-gateUpstream` method of the transport resources.

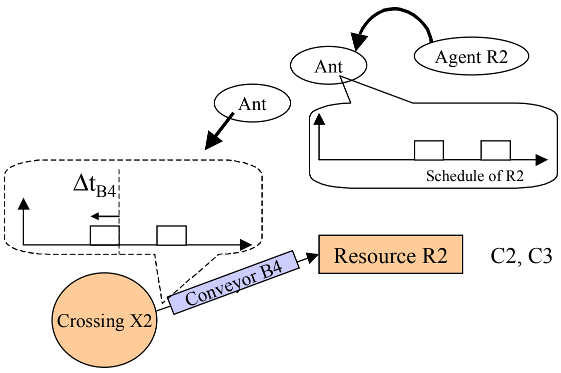


Fig. 2. Propagation of a load forecast.

5.3 Discussion

Summarizing, the downstream propagation of intentions enables the construction of local load forecasts, whereas the upstream propagation of these local forecasts creates opportunity forecasts for remote resources that can be used locally to guide routing decisions. Moreover, nowhere are individual agents exposed to the complexity and the dynamics of the overall system. Like the ants, the manufacturing control agents operate locally and the overall behavior emerges.

A difference between the two propagations is that intentions follow a single route through the system; the mobile agents from order agents choose a route at every decision point. In contrast, forecasts spread out across the system when information arriving at an exit is propagated to all entrances; the mobile agents clone themselves when there is more than one entrance.

6. Decision Taking

The order agents make decisions based on the locally available information. First of all, the agents respect all the hard constraints and eliminate options that are closed to them. Next, they use a decision rule that takes into account the information on the local blackboards. These rules should be randomized to ensure exploration and to avoid getting stuck in a pathological pattern—think about two persons, arriving from opposite sides, that want to pass each other and that start changing sides in a synchronized pattern, thus staying in front of each other all the time. Evidently, the higher is the perceived benefit, the more likely will an order agent select an option.

These decision rules are likely to remain the most maintenance-prone part of the manufacturing control system. Indeed, this is the part where the user may incorporate system-specific and situation-specific knowledge. Fortunately, it is the top layer and there will be no cascade of software maintenance task caused by this. Note that the services of staff agents can be used in these decision rules.

Importantly, the decision rules of the agents must account for the fact that other agents base their decisions on declared intentions. In addition, the propagated information only becomes available with some delay. Because of this, proper precautions are needed to avoid that the system behaves too nervously. Indeed, if every order agent (or his mobile agent) changes his intentions immediately whenever he perceives a better opportunity, the system is unlikely to achieve its goal (smooth and optimized production). For the forecasts to have any value, the intentions of only a small percentage of the order agents may change before the other agents are able to observe this within the updated forecast—recall that the agents regularly refresh their information while the old information evaporates. Therefore, order agents and their mobile agents must have a build-in tendency to stick to their intentions, which they declared earlier:

- The perceived utility increase has to pass a threshold value before an agent will change his intentions. If it requires them to change their mind, agents do not go for the highest perceived value until the benefit becomes significant.
- This threshold is higher when an agent recently changed his intentions. Agents limit and randomize the frequency at which they change intentions.
- This threshold increases when the moment of executing the intentions comes nearer. Agents are more relax about changing their intentions when the other agents still have plenty of time to adapt to these changes.

Overall, such behavior is very common in smoothly operating and well-behaved human organizations.

Finally, note that the novelty and contribution of the agent-based design comes foremost from the system engineering properties (desirable overall behavior emerges from putting together agents with limited exposure) rather than from clever decision methods. The approach presented in this paper helps to cope with the need to rapidly develop adequately performing control systems that can adapt to disturbances and changes while being operational, rather than developing the optimal solution for a static situation.

7. Conclusions

This paper discusses the development of multi-agent coordination and control systems based on techniques inspired by biological system, i.e. food foraging ants. Moreover, it identifies the key achievement of this biological example: limited exposure of the individuals combined with the emergence of robust and optimized overall system behavior. Furthermore, the paper pinpoints essential properties of the ant solution:

- The environment is a part of the solution, shielding the remainder of the system from its complexity and dynamics.
- Global information is made available locally. On its way through the system, this information is transformed, in appropriate manners, to enable the agents to make local decisions based on locally available information while being aimed at global goals (e.g. reaching a processing unit along an optimized path).

These two properties must be combined with better-known elements, like positive feedback—required for the emergence of order—and randomized decision taking—required to explore and to avoid pathological behavior.

This results in an approach to build multi-agent coordination and control systems comprising the following steps:

- Agentify the environment to make it part of the solution. Apply the PROSA reference architecture to separate resource, logistic, and process concerns.
- Develop the feasibility control layers.
- Develop the operational control layers.
- Optionally, add staff layers.
- Develop and tune the decision mechanisms in the agents.

In this development process, only the last step addresses decision taking. The remainder reflects facts about the system with the exception of situations where this would imply a combinatorial explosion (where an approximated solution must be used, possibly involving lazy evaluation).

Also, these steps need not be executed sequentially. The design is capable of evolving gradually into higher levels of sophistication as well as tracking changes in the underlying ironware system.

The paper reports on developments in the area of manufacturing control. However, the proposed solutions are not specific to manufacturing. The assumptions on which the solutions rely are present in most situations where macroscopic ironware is controlled by a multi-agent system running on a computer network. The following requirements are important:

- The underlying ironware system does not compete with the agent system for computer and communication resources.
- The agent system executes much faster than the underlying ironware system (order of magnitude difference in bandwidth).
- The underlying ironware system is subject to physical laws and, virtually invariant, technological constraints.
- Small enhancements of the performance of the underlying system along vital performance criteria pay back the computer network on which the agents live (e.g. increase a throughput of 1000 cars per day by 1%).

This niche for multi-agent control makes it possible to have the control system run ahead in time. Based on the propagation of the intentions of agents, forecasts are constructed. These forecasts are propagated in turn to enable the decision takers to revise and optimize their decisions/intentions. Note however that proper measures will be required to dampen the system and keep the forecasts sufficiently valid.

Finally, the design is based on coordination mechanisms that avoid blocking rendezvous interactions amongst the agents. Agents observe and act; there are no timeouts that require tuning in this system. This design is capable of addressing a lot of small issues simultaneously, in contrast to designs based more direct and explicit negotiation amongst the agents.

Acknowledgements

This paper presents work funded by the Research Fund of the K.U.Leuven and the European Commission.

References

1. Grassé, P.P., La theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs, Insectes Sociaux 6 (1959)
2. Theraulaz, G., A brief history of Stigmergy, Artificial Life 5 (1999) pp. 97-116
3. Ossowski, S., Co-ordination in Artificial Agent Societies – Social Structures and its Implications for Autonomous Problem-Solving Agents, Springer-Verlag, Berlin (1999)
4. Parunak, H.V.D., Go to the Ant: Engineering Principles from Natural Multi-Agent Systems, Annals of Operations Research (1997)
5. Parunak, H.V.D., Baker, A.D., Clark, S.J., The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design, in Proc. 1st Intern. Conf. on Autonomous Agents, Marina del Rey, CA (1997)
6. Baker, A. D., Metaphor or reality: A case study where agents bid with actual costs to schedule a factory. In S. Clearwater (ed.), Market-Based Control: A Paradigm for Distributed Resource Allocation. World Scientific, River Edge, NJ (1996)
7. Peeters, P, Van Brussel, H., Valckenaers, P., Wyns, J., Bongaerts, L., Kollingbaum, M., Heikkilae, T., Pheromone-based Emergent Shop Floor Control Systems for flexible Flow Shops, IWES'99, Kobe (1999).
8. Di Caro, G., Dorigo, M., AntNet: Distributed Stigmergic Control for Communication Networks, Journal of Artificial Intelligence Research 9 (1998).
9. Dorigo, M., Bonabeau, E., and Theraulaz, G., Ant Algorithms and Stigmergy, Future Generation Computer Systems, 16 (2000), No.8, pp. 851-956
10. Dorigo, M., Di Caro, G., The Ant Colony Optimization Meta-Heuristic. In: New Ideas in Optimization, D. Come, M. Dorigo, F. Glover (Eds.), McGraw-Hill, London (1999) pp.11-32
11. Kollingbaum, M., Heikkilä, T., Peeters, P., Matson, J., Valckenaers, P., McFarlane, D., Bluemink, G-J., Emergent Flow Shop Control based on MASCADA Agents, MIM 2000, Patras, Greece (2000)
12. Snyers, D., Cooperative Agents for Dynamic Routing in Communication Networks and Job Shop Scheduling (1998)
13. Wyns J., Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration, Ph.D. thesis K.U.Leuven (1999) ISBN 90-5682-164-4
14. Van Brussel, H., Jo Wyns, Paul Valckenaers, Luc Bongaerts, Patrick Peeters, Reference Architecture for Holonic Manufacturing Systems: PROSA, Computers In Industry, Vol. 37 (1998) pp. 255-274
15. J. Peppard, Benchmarking Business Processes: a Framework and Classification Scheme, Proc. Esprit-Copernicus Symposium, Wroclaw, Poland, 23-24 April 1998, pp. 213-240.
16. Valckenaers P., Van Brussels H., Brueckner S., Wyns J., Peeters P., Deadlock Avoidance in Flexible Flow Shops with Loops, 6th IFAC Workshop on Intelligent Manufacturing Systems, 24-26 April 2001, Poznan, Poland (2001)

Virtual Enterprise Modeling and Support Infrastructures: Applying Multi-agent System Approaches

Luis M. Camarinha-Matos [1] and Hamideh Afsarmanesh [2]

[1] New University of Lisbon, Faculty of Sciences and Technology
Quinta da Torre, 2825 Monte Caparica, Portugal
cam@uninova.pt
[2] University of Amsterdam, Faculty of Computer Science
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
hamideh@science.uva.nl

Abstract. Virtual enterprises paradigm represents an important application field for multi-agent approaches, both in terms of modeling and infrastructure development. This article summarizes the main challenges in this field and describes several current Multi-Agent System application approaches. A particular emphasis is given to the creation and operation phases of the virtual enterprise life cycle. Several open challenges in this area are also introduced.

1. Introduction

Advances of the last decades in information and communication technologies have triggered fast developments in a large number of application areas involving cooperation among networked entities. On one hand, the Internet developments enabled the implantation of a number of already foreseen paradigms that were just waiting for the proper conditions, and on the other hand they motivated the development of new concepts.

In this context a large number of new terms have populated the scientific and technological literature such as the e-commerce, e-business, virtual organizations, virtual enterprises, networked enterprises, virtual communities, communities of practice, and even e-manufacturing. As usual in times of rapid advances, this fast proliferation of new terms does not always represent well-understood or truly new concepts. Nevertheless one of the main underlying trends in research and development is the focus on models, protocols, and mechanisms to support the collaboration of pre-existing entities in distributed environments, be it among organizations, among people, or among people and organizations.

Cooperation in this context involves heterogeneous, autonomous, and not totally cooperative entities. Some related issues have gained a particular importance, such as interoperability and information sharing, common ontologies, safe communications, privacy and authentication, and base support infrastructures. But cooperation in a networked environment is not purely a technological issue. There is also a need to identify and support the necessary re-organization, handle the social and legal impacts, redefine roles and corresponding skills for the human actors, etc. In other words, there is a strong requirement for a multidisciplinary approach.

This article discusses one facet of this new trend, namely the virtual enterprises paradigm, and in particular focuses on the use of multi-agent approaches to model and support such cooperative organizations.

In the remaining of this chapter, first in section 2 the concept of virtual enterprise and related terminology are presented, as well as the requirements for a supporting infrastructure; then in section 3 the application of multi-agent systems (MAS) approaches to the VE creation is described; section 4 discusses some relevant issues of MAS support for VE operation, namely in the areas of advanced information management and remote supervision; finally section 5 draws some conclusions and a summary of open challenges.

2. The Virtual Enterprise Paradigm

2.1 Virtual Enterprise and Related Concepts

VE concept. The virtual enterprise (VE) research represents a growing and multidisciplinary area that still lacks a precise definition of the concepts and an agreement on the used terminology. So far, there is no unified definition for this paradigm and a number of terms are even competing in the literature while referring to different aspects and scopes of virtual organizations [8], [9], [18]. Nevertheless a commonly used definition is [8]:

"*A virtual enterprise is a temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks.*"

Two key elements in this definition are the *networking* and *cooperation*. Clearly, there is a tendency to describe a virtual enterprise as a network of cooperating enterprises. A number of pre-existing enterprises or organizations with some common goals come together, forming an interoperable network that acts as a single (temporary) organization without forming a new legal entity nor establishing a physical headquarter. In other words, Virtual Enterprises materialize through the integration of skills and assets from different firms into a single business entity. This paradigm challenges the way companies are organized and managed. Companies, especially the Small and Medium size Enterprises (SMEs) must join skills and resources in order to survive and gain competitive advantage in a global market environment.

The wide variety of different networked organizations and the emergence of new product and service provision organizational paradigms have led to the introduction of a number of related terms such as (Fig. 1):

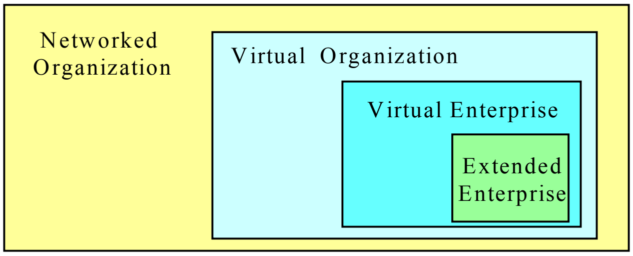


Fig. 1. Examples of networked organizations

Extended Enterprise. This term, the closest "rival" to virtual enterprise, is better applied to an organization in which a dominant enterprise "extends" its boundaries to all or some of its suppliers. This is a situation typically found in the automotive industry and some other stable supply chains.

Virtual Enterprise. In comparison to Extended Enterprise, the VE can be seen as a more general concept including other types of organizations, namely a more democratic structure in which the cooperation is peer to peer. In this sense, an extended enterprise can be seen as a particular case of virtual enterprises.

Virtual Organization. This is a concept similar to a virtual enterprise, comprising a network of organizations that share resources and skills to achieve its mission / goal, but not limited to an alliance of enterprises. An example of virtual organization could be a virtual municipality organization, associating via a computer network, all the organizations involved in a municipality (e.g. city hall, municipal water distribution services, internal revenue services, public leisure facilities, cadastre services, etc.). A virtual enterprise is, therefore, a particular case of virtual organization.

Networked Organization. This is perhaps the most general term referring to any group of organizations inter-linked by a computer network, but without necessarily sharing skills or resources, or having a common goal. Typically, networked organizations correspond to a very loose type of organization.

Life cycle phases. A VE evolves along various stages along its life cycle including: the creation, operation, evolution, and dissolution stages (Fig. 2) [8], [9].

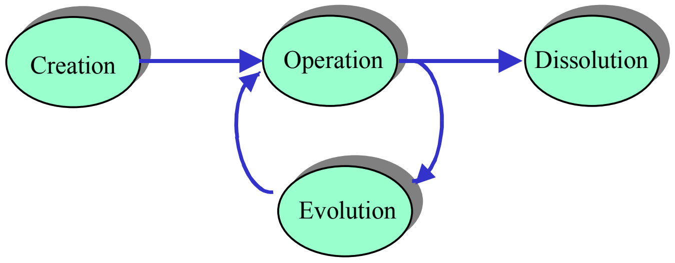


Fig. 2. Life cycle of a VE

i. *Creation*. This is the initial phase when the VE is created / configured and for which some of the major required functionalities are: Partners' search and selection, Contract Negotiation, Definition of access rights and sharing level, Join / Leave procedures definition, Infrastructure configuration, etc.
ii. *Operation*. This is the phase when the VE is performing its business process(es) in order to achieve its common goal(s), and which requires functionalities such as: Basic secure data exchange mechanisms, Information sharing and visibility rights support, Orders management, Distributed and dynamic planning and scheduling, Distributed task management, High levels of task coordination, Collaborative engineering support, etc.
iii. *Evolution*. Evolutions might be necessary during the operation of a VE when it is necessary to add and / or replace a partner, or change roles of partners. This

need might be due to some exceptional event, such as (temporary) incapacity of a partner, changes in the business goal, etc. Functionalities similar to the ones specified for the creation phase are necessary to also be supported here.

iv. *Dissolution.* This is the phase when the VE finishes its business processes and dismantles itself. Two situations may be the cause for VE dissolution, either the successful achievement of all its goals, or by the decision of involved partners to stop the operation of the VE. The definition of liabilities for all involved partners is an important aspect that needs to be negotiated. For instance, the responsibility of a manufacturer more and more remains during the life cycle of the produced product till its disassembly and recycling.

Participant Roles. An enterprise may play different roles within a VE during its life cycle. In other words, several kinds of actors can be found in and around a VE environment, acting as: the VE Coordinator, VE member, Network directory node, Broker, etc:

– *VE Coordinator.* The VE Coordinator will be the regulator component of the VE related activities. The coordinator is either a node specialized in coordination and added to the VE-network, or its role can be played by an already existing VE-member. In addition to the VE coordination role, responsible for the global goal, other enterprises may assume the role of coordinators of sub-processes that might be decomposed and performed by a sub-consortium of enterprises.

– *Member Enterprise.* Enterprises with different skills and/or capacities participating in a VE constitute the Member Enterprise nodes.

– *Network Directory node.* One or more nodes in a network of enterprises may act as the directory nodes. Here the network refers to a general wide area network such as Internet to which a large number of enterprises have access or a closed community of enterprises that establish long-term cooperation plans (industry cluster). Various VEs may coexist in this network and clearly a node in the network may belong to several VEs.

– *Broker / initiator.* This is the role played for instance by a company (not necessarily the VE coordinator) that initiates / creates a VE, plans its business process, and searches for partners.

Lack of a reference model. The materialization of this paradigm, although enabled by recent developments in communication technologies, computer networks, and logistics, first requires the definition of a suitable reference model for cooperation and the development of a flexible supporting platform and second the development of appropriate protocols and mechanisms. In spite of the many efforts being put in this area, the current approaches and the developed experimental prototypes are quite limited, still lacking a comprehensive and flexible characterization of the multiplicity and variety of the cooperation scenarios.

2.2 Variety of Classes of Virtual Enterprises

Classification facets. A large number of diversified networked organizations of enterprises fall under the general definition of VE, requiring diversified panoply of supporting functionality. There is clearly a need to first classify different VE paradigms in terms of their characteristics and respective requirements, before the paradigm can be properly addressed and modeled. A simplified taxonomy of virtual enterprises was proposed in [8], [9]:

– *Duration.* Some alliances of enterprises are established towards a *single business* opportunity, and are dissolved at the end of such process. This situation corresponds perhaps to the most typical kind of virtual enterprise, for which examples can be found in large scale engineering systems, such as, a construction consortium involved in building a one of a kind bridge or a railway. But there are also *long term alliances* that last for an indefinite number of business processes or for a specified long-term time span. The first case raises the need for an infrastructure supporting very dynamic consortium creation / dissolution. In the second case however, the emphasis is put on the operation of the VE and on the support for dynamic business process definition and supervision.

– *Topology.* According to the topology of the network, there are situations that show a *variable / dynamic nature*, in which some enterprises (non strategic partners) can dynamically join or leave the alliance according to the phases of the business process or other market factors. But in many sectors there exist established supply chains with an almost *fixed structure* (little variation in terms of suppliers or clients during the VE life cycle). Another possibility to be considered is the temporary interaction with other enterprises that do not belong to the VE, such as the occasional suppliers or the spontaneous clients (for instance via the electronic commerce mechanisms).

– *Participation.* Another facet to be considered is the possibility of either an enterprise participating simultaneously in *multiple alliances*, or being dedicated to a *single alliance* (exclusivity membership). In the non-exclusive case, the supporting infrastructure must handle various VE participation spaces and to cope with strict cooperation and information visibility rules, to preserve the requirements of every individual enterprise.

– *Coordination.* In terms of the network coordination, various approaches can be found. In some sectors, as typified by the automobile industry, there is a dominant company "surrounded" by a relatively fixed network of suppliers (*star-like* or *centralized coordination structure*). The dominant company defines "the rules of the game" and imposes its own standards, namely in terms of the business process models, information exchange mechanisms and access rights, on the others. The concept of extended enterprise can be used to describe this particular case, as it represents a dominant enterprise extending its borders over the satellite suppliers and service providers. A different organization can be found in some supply chains, without a dominant company (*democratic alliance*). In such networks all the nodes cooperate on an equal basis, preserving their autonomy, but joining their core competencies. But even in this case, a coordinator node is necessary in order to administer the general information regarding the VE membership, and to monitor the organizational structure and joint cooperation principles. In an extreme case, once a successful alliance is formed, companies may realize the mutual benefits of joint management of resources and skills and they may tend to create a kind of joint coordination structure (*federation*).

Visibility scope. Both related to the topology and coordination is the aspect of visibility scope, i.e., “how far”, along the network, can one node “see” the VE configuration. In most cases a node only sees its direct neighbors (suppliers, clients) (*single level visibility*). That is the case observed in most supply chains. In more advanced coordination situations however, a node might have some visibility rights over other (non-directly related) enterprises (*multilevel visibility*), including some levels of information access visibility, which may lead to a more optimized operation of the VE. Furthermore, monitoring of order fulfillment, planning, scheduling / rescheduling, workload distribution, and optimized resource management are examples of advanced task supervision and VE coordination that require an extensive visibility scope. Typically visibility scopes are bilaterally agreed between enterprises, or generally agreed through the VE enforced contracts among all VE-involved members and the VE coordinator.

2.3 Application Examples and Infrastructure Requirements

Examples. A typical application area for the VE paradigm is the industrial manufacturing. Nowadays, most of the manufacturing process is not carried on by a single enterprise anymore. Companies feel the need to focus on their core competencies and join efforts with others, in order to fulfill the requirements of the new products and associated services demanded by the market and complying with more demanding quality standards and environment regulations. In a VE every enterprise is just a node that adds some value to the process. Although most classic examples of cooperative networked organizations can be found in some particular business domains such as the automotive industry, this tendency is spreading to many other areas including the food and agribusiness industry [11], civil engineering [37], shipbuilding, electronics [5], etc.

Similar to the manufacturing industries, the need to remain competitive in the open market also forces the service providing companies to seek “world class” status. That is, for instance, the case of Tourism industry, where companies remain focused on their core competencies and realize the need to look for alliances when additional skills / resources are needed to fulfill business opportunities. Cooperation among the actors / entities in Tourism industry is not a new phenomena. For instance, travel agencies typically offer aggregated or *value-added-services* (VAS) composed of components supplied by a number of different organizations. For instance, to provide a “book a complete journey plan” service that may include several means of traveling, several hotel bookings, car rentals, leisure tour bookings, etc., a networked cooperation must exist among many different organizations [2]. The insurance and consultation sectors are other examples for which VEs have been developed.

As presented above, such types of cooperative networks can be extended to other organizations besides enterprises. For instance, a virtual organization can involve the administrative bodies of a municipality such as the city hall, internal revenue services, water distribution services, cadastre services, etc, giving the appearance of a single organization to their customers, the citizens. Another example could be a virtual university where different educational institutions join efforts to offer, preferably via Internet, a joint course combining the best expertise of each institution. Similar organizations can be foreseen in the health care and elderly care sectors.

Infrastructure requirements. In terms of infrastructure there is a need for an extensive list of tools to support the various phases of the VE life cycle [8], [9].

Creation/configuration phase of a VE. The creation/configuration phase of a VE needs tools for partners selection and for decision support to help the negotiation process and all the dynamics associated to the joining / leaving of enterprises. Examples of such tools include: *Partners search and selection, Business process planning, Tender formation, Contract negotiation, awarding and management.*

From an "administrative" point of view, there is a need for *configuration tools* to help in the set up of the infrastructure, according to the agreements made during the contractual phase. For instance, it is necessary to configure the infrastructure to accommodate the particular set of messages agreed between every two nodes of the VE, or the specific information sharing and visibility rights for VE members.

Operation phase of a VE. After a VE is established and started its operational phase various levels of interactions among its members must be supported. The main functionalities include:

i. *Basic information exchange interactions.* The minimal level of services required in a VE supporting platform are the following: Information exchange mechanisms to support the exchange of shared/public commercial data (e.g. contract-related interactions), technical data (e.g. product models and quality information), general information (e.g. market statistics and catalogues of products/services), etc. At this level it is necessary to support interoperability between standards (e.g. EDIFACT and STEP). Support for safe communications, authentication of interlocutors and auditing mechanisms are required.

ii. *Events / exception handling.* As a member of a VE, an enterprise acts as an event driven system. Namely, it needs to handle asynchronous events and exceptions either generated inside the company or by other nodes in the network.

iii. *Advanced coordination.* To properly support the functionality of a VE, and independent of the size of the VE, there is a need for a VE coordinator. The main task of the VE coordinator is the monitoring of the job status (distributed business process) and comparing it to the VE plans as described in the contracts. In the case that an enterprise fails to perform its duties, the VE must be reconfigured to replace the failing enterprise with another one. To support this functionality it is also convenient to have a distributed business process planning / modeling tool that allows for re-planning and re-scheduling of business processes.

iv. *Material / services related aspects*. It is important to support functionalities necessary to represent and monitor the flow of products and services through the VE network. Some required functionalities include: Materials/services-flow management; Logistics planning and management; Forecasting; Specific information flows related to product (bar coding, POS).

v. *Collaborative environments.* New business practices require tools to support collaborative activities such as Concurrent Engineering or collaborative problem solving / consultancy with teams composed of human experts belonging to different VE members.

Dissolution of a VE**.** This is the least studied phase of the VE life cycle in the current literature, but some tools are certainly required to support the following main aspects among others:

- Definition of general liabilities upon the dissolution of the VE;
- Keeping track of the individual contributions to a product / service that is jointly delivered; namely in terms of the quality and product life cycle maintenance;
- Redefinition of information access rights after ceasing the cooperation; and
- Assessing the performance of partners, generating information to be used by partners' selection tools in future VE creation.

New emerging services. It is important to notice that the VE paradigm is not an isolated phenomenon. Many new other services are rapidly being proposed over the Internet and some of them contribute to the functionalities required for virtual enterprises. One important related area is the Electronic Commerce that proposes solutions on important issues such as: Organization and publication of electronic catalogues and related mechanisms; Security mechanisms, namely to support interchange of payments related information; Advanced and customizable search engines, some of them based on mobile software agents; service publication and discovery tools; and Legal issues related to electronic-based business transactions.

Legacy systems. Finally it is important to mention that coping with legacy systems is a mandatory requirement for any VE-supporting infrastructure. The strong reliance on standards will contribute to facilitate the interfacing of existing applications with the VE infrastructure, but unfortunately not all classes of information that need to be exchanged among VE nodes are covered by existing standards. Initiatives of groups of application developers such as the Workflow Management Coalition [36] or the Open Applications Group [23] can also contribute to facilitate this process. In general, it is necessary to develop some interface / mapping layer, at each enterprise, to adequately have this enterprise interacting with the VE, via the VE infrastructure.

It shall also be noticed that most legacy applications were designed for a local operation (enterprise-centered) and to be operated by humans. In order to have these applications supplying information to or consuming information from the VE network, it is clearly necessary to extend their functionality.

Interoperability among enterprise applications represents a major challenge for supporting the rapid formation of VEs, in response to new business opportunities. On the other hand it is important to have in mind that each enterprise has its own culture and way of doing business. Furthermore the level of information sharing among VE members is likely to evolve, either with the change of the trust level among partners or with the evolution of the VE configuration in time. Therefore, flexible VE-related coordination and information visibility rights definition mechanisms are needed, in order to support both the autonomy and evolution in behavior of the VE members.

Enabling technologies and standards. The emergence of a number of standards and technologies represent potential enabling factors, such as for instance:

– Open interoperable underlying network protocols (TCP/IP, CORBA-IIOP, HTTP, RMI, SOAP),
– Open distributed object oriented middleware services (Java2 Framework, CORBA Framework, ActiveX Framework),

- Standardised modelling of business components, processes and objects (EJBs, OAG and OMGs Business Objects and Components),
- Business Process Modelling Tools and Languages (UML, UEML, PIF, PSL, WfMC XML-based Business Language)
- Open and standard business process automation and Workflow Management Systems (WfMC, OMG-JointFlow, XML-WfMC standards, many commercial products),
- Standard interfacing to federated multi-databases (ODBC, JDBC),
- Intelligent Mobile Agents (FIPA, OMG-MASIF, Mobile Objects, Java Initiatives),
- Open and standard distributed messaging middleware systems (JMS, MS-Message Server, MQSeries, FIPA-ACC),
- XML-Based E-Commerce Protocols (BizTalk, CBL, OASIS, ICE, RosettaNET, OBI, WIDL),
- Web Integration Technologies (Servlets, Java SP, MS-ASP, XSL).

However, several of these technologies are in their infancy and under development, requiring considerable effort to implement and configure comprehensive VE/VO support infrastructures.

Therefore, although the advantages of the Virtual Enterprise are well known at the conceptual level [8], [18], the practical implantation is still far from the expectations, except for the more stable, long-term networks, or supply chains. The potential agility of a VE in terms of fast reaction to business opportunities (opportunistic VE) is certainly a desirable feature in a scenario of fast changing market conditions, but the early phase of VE planning and creation is still a difficult one that needs to be adapted even by advanced and competitive enterprises. Some of the obstacles include the lack of appropriate support tools, namely partners search and selection, VE contract biding and negotiation, competencies and resources management, well-established distributed business process management practices, task allocation, performance assessment, inter-operation and information integration protocols etc. Further problems include the lack of a common ontology, and the proper support for socio-organizational aspects e.g. lack of a culture of cooperation, the time required for trust building processes, need for BP reengineering and training of people, etc. There is also the fact that the fast evolution of the information technologies often represents a disturbing factor for non-IT companies.

2.4 MAS and Virtual Enterprises

There are a number of characteristics in the VE domain that make it a suitable application area for MAS. Examples of such characteristics include:

- A VE is composed of distributed, heterogeneous and autonomous components, a situation easily mapped into MAS.
- Coordination and distributed problem solving also tackled by MAS are critical problems in VE management.
- Decision making with incomplete information, and involvement of network members as autonomous entities, that although willing to cooperate in order to reach a common goal might be competitors regarding other business goals, is another common point.

- The effective execution and supervision of distributed business processes requires quick reactions from enterprise members. Computer networks being the privileged media for communication, there is a need for each company having a "representative" in (or "listening" to) the network. This can be supported by agents.
- Recent developments in VE are changing the focus from information modeling and exchange to role modeling, addressing aspects of distribution of responsibilities, capabilities and knowledge.
- The phase of VE formation in which it is necessary to select partners and distribute tasks, shows market characteristics and negotiation needs that have been research issues in MAS.
- A VE consortium is a dynamic organization that might require re-configurations – e.g. replacement of partners, changes in partners' roles, etc., for which a flexible modeling paradigm is necessary.
- VE supporting functionalities need to interact with the "local" environment (legacy applications and humans).
- The scalability property of MAS seems particularly adequate to support dynamic VEs in which different levels of cooperation with different sets of partners might be established at different phases. On the other hand, each enterprise might itself be seen as composed by a network of semi-autonomous entities (departments).
- More flexibility than in a client-server model is required to support dynamic change of roles of the VE members.
- Continuous evolution of business models, technologies, organizational paradigms, and market conditions require effective support for evolution and a high level of modularity of the infrastructures.
- New forms of teamwork, namely cooperative concurrent engineering, are emerging in the context of VEs.
- Finally there is a need to handle the requirements of autonomy vs. cooperative behavior for which federated MAS approaches may provide a balanced solution.

In spite of these positive arguments in favor of the use of MAS in VE, there are also some obstacles. MAS is still lacking some important characteristics that represent inhibiting factors for its application in real world VEs:

- Robustness of development environments.
- Easy interface with legacy systems.
- Security mechanisms and virus protection.
- Standards and common ontologies to support interoperability.
- Culture interchange between AI and BP communities.
- Realistic demo cases.

It is also important to notice that MAS has to face the competition of other traditional or emerging approaches: component-based software; loosely coupled transactional systems; service federation and service markets.

3. Virtual Enterprise Creation

3.1 Partners Search and Selection

The search and selection of partners is a very important process in the life cycle of a VE. The need for this process comes first in the creation phase. When a new business opportunity is detected, the initiator of the VE has to look for the most suitable partners for the new consortium. Furthermore, during the normal operation of a VE it might be necessary to find suppliers for a particular component or service not offered by other members or even to replace a member that founds itself unable to fulfill its commitments.

Recently there has been a considerable effort put in the so-called electronic procurement. The main objectives in this area include the definition of "normalized" procedures for public announcement of business offers, reception, and management of bids. Standardization is in fact the main obstacle in electronic procurement. The VE partners' search and selection activity shares several similarities with the classic electronic procurement. Both areas require the identification of potential suppliers / partners to be addressed, the adoption of normalized specification of requirements and bids, management of directories of potential partners, management of bids, and decision support functionalities.

A growing number of works are being published on the application of multi-agent systems and market-oriented negotiation mechanisms for the VE formation. One such example can be found in [30]. This work assumes a virtual market place where enterprises, represented by agents that are geographically distributed and possibly not known in advance, can meet each other and cooperate in order to achieve a common business goal. A MAS architecture is proposed to model the electronic market to support the formation of the VE. In addition to the agents representing the enterprises, there is a market agent – coordinator or broker – that is created and inserted in the MAS community when a business opportunity is found.

A multi-round contract-net protocol is followed: the market agent sends invitations to the electronic market corresponding to each of the VE sub-goals; receives bids and evaluates them; the most favorable ones are selected based on a multi-criteria mechanism and constraint-based negotiation. Examples of considered criteria are lower-cost, higher quality, higher availability, etc. Utility values are associated to each of these criteria and a linear combination of attribute values weighted by their utility values is used. Multiple negotiation rounds can take place. At the end of each round bidders receive indication whether their bids are wining or loosing and a rough qualitative justification, allowing them to change the parameters of their proposals.

A similar work is found in [22] where a more detailed analysis of the problem of goal decomposition, leading to a hierarchy of VE goals, is done. In addition to the enterprise agents and VE coordinator agent (broker), an information server agent is introduced to keep public information related to common organizational and operational rules, market environment, enterprises and products / services provided, etc. The need for a common ontology to support the communication among agents is explicitly introduced and a multi-attribute, constraint-based negotiation / selection process is implemented.

In [16] there is a proposal to use mobile agents that are sent to potential suppliers to check their competencies. These agents make an on-site broad selection (rough qualitative analysis), while a fine evaluation with the information brought back by them is then performed at the sender's place. As part of the selection process, an assessment of the partnership performance of the candidates, based on their history of cooperation, is also made.

The work described in [33] identifies the need for yellow pages agents that are responsible to accept messages for registering services (similar to the information agent server mentioned above). They also consider the concept of Local Area, a quasi-physical division of the network that can be controlled by a local area coordinator. This is a similar concept to the Local Spreading center first introduced by the HOLOS system [27].

Finally [19] elaborates further on the application of market-oriented principles, with particular reference to the principles of general equilibrium in micro-economics.

These proposals are limited by a number of factors which affect their practical implantation including:

– Lack of common standards and ontologies, a situation difficult to overcome in a general "open universe" of enterprises.
– None of these proposals takes into account more subjective facets like trust, commitment, successful cooperation history, etc.
– In general they pay little attention to the implantation aspects and the management of the yellow pages / market place.
– Security issues in the negotiation process are not addressed, a critical point as the agents are only partially cooperative (they might be self-interested, competitive, and even exhibit antagonistic behavior).
– The attempt to reach a fully automated decision-making process, although an interesting academic exercise, is quite unrealistic in this application domain.

On the other hand, as agents are designed and developed independently, it is quite difficult to guarantee coordination unless common rules ("social laws") are adopted.

3.2 Cluster and VE Formation

One approach to overcome some of the mentioned drawbacks is to consider the partners search and selection within a long-term industry cluster. The concept of **cluster of enterprises**, which should not be confused with a VE, represents a group or pool of enterprises and related and supporting institutions that have the potential, and the will, to cooperate with each other through the establishment of a long-term cooperation agreement. For each business opportunity found by one of the cluster members, a subset of the cluster enterprises may be chosen to form a VE for that specific business opportunity. The more frequent situation is the case in which the cluster is formed by organizations located in a common region, although geography is not a major facet when cooperation is supported by computer networks.

The cluster enterprises are normally "registered" in a directory, where their core competencies or offered services are "declared". Based on this information, the VE initiator / creator can select partners when a new business opportunity is detected. Clearly, several VEs can co-exist at the same time within a cluster, even with some members in common. A cluster represents a long-term organization and therefore, an adequate environment for the establishment of cooperation agreements, common infrastructures and ontologies, and mutual trust, which are the facilitating elements when building a new VE (a kind of controlled marketplace). The cluster does not need to be a closed organization; new members can adhere but they have to comply with the general operating principles of the cluster. For the formation of a VE, preference will be given to cluster members but it might be necessary to find an external partner in case some skills or capacities are not available in the cluster. The external partner will naturally have to adhere to the common infrastructure and cooperation principles. In addition to enterprises, a cluster might include other organizations (such as research institutions, sector associations, etc.) and even free-lancer workers. The establishment and management of clusters through adequate infrastructures, represents therefore an important support for the creation of agile virtual enterprises.

An example of a MAS application to VE creation in the context of an industry cluster formed by twelve companies in the domain of moulds and die-casting can be found in [29]. The cluster is legally represented by a broker entity that has an expert responsible for getting and analyzing business opportunities. By means of a *broker agent* an opportunity is transformed in a distributed business process that is then distributed to the (potential) enterprises within the cluster. In the end of the whole process, a set of possible teams of enterprises ("potential" VEs) that can carry out that opportunity is formed and the most suitable team is proposed (but the ultimate decision is made by the human experts). Figure 3 illustrates the formation of a set of teams of enterprises within the cluster to attend a given distributed business process. In this example, there are three VEs capable of accomplishing the business process but VE1 was the selected team.

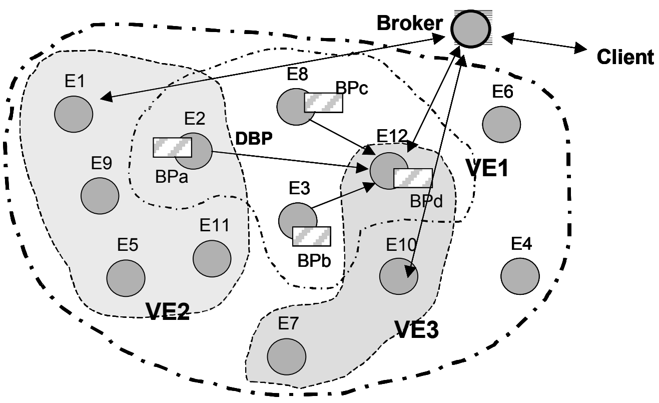


Fig. 3. Multiple VE hypothesis within a Cluster

A multi-agent-based system – the MASSYVE Mould Broker System (MMBS) – was developed to support the cluster's human broker in the management and decision-making process of selecting the most adequate consortium of enterprises that can satisfy a given business opportunity. In the case that no partners are found in the cluster, an additional partners' search and selection tool is called to find potential candidates external to the cluster, based on directories of enterprises available on Internet [10].

The following macro actions are considered in the selection process:

i) For each business opportunity, the broker gets, analyzes and distributes the client order to the enterprises whose competence fits the tender;

ii) The involved enterprises get, analyze, and bid in the case they are interested or are capable to attend the client order's requirements. A bid preparation involves a direct intervention of the enterprise's manager so that he/she can indicate its price and eventually refine the proposed delivery date. Once the bid is sent, the enterprise makes a conditional booking in its agenda for that business until it receives the final result;

iii) Based on the positive bids, the broker identifies several alternative VEs;

iv) If no solution is generated with the cluster members bids, an additional partner search tool is executed in order to find some other enterprise(s) – out of the cluster – that can fulfill the order's requirements;

v) A VE is finally selected (with human help) and created, and the involved enterprises are noticed about the final result.

In this prototype the following agent types are considered:

– Mould-Broker agent: it is the global system supervisor, acting as the interface between the system and the human broker. There is one – and only one – mould-broker agent in a particular system. It can interact with the *facilitator* and *consortium* agents.
– Enterprise-Agent: it represents a given enterprise, member of the cluster. There are as many enterprise agents as existing enterprises in the cluster. An *enterprise-agent* can interact with the *consortium* and *facilitator* agents.
– Facilitator agent: it represents a partition of the cluster members according to a particular competence. There are as many facilitator agents in a particular system as existing competence views. The mould-broker agent first sends a tender to the facilitators whose competence fits the client's order *type*. This speeds up the contract process as well as avoids the unnecessary message exchange among non-potential bidders. The *facilitator* can interact with the *mould-broker*, the *consortium,* and *enterprise* agents.
– Consortium agent: it is a temporary agent created to manage the process of generating a VE alternative for a given business opportunity, based on the bids received from the enterprise-agents. For each business opportunity under analysis there will be as many consortia agents as feasible VE consortia. Once the broker and the responsible humans decide for the best solution and award the contract to the involved enterprise-agents, the respective consortium agent assumes a supervision role regarding the business process execution and the other consortia dismantle themselves.

3.3 Organizational Forms

In addition to partners selection, it is important to define the organizational model of the consortium and the types of cooperation relationships that are typically regulated by contracts / cooperation agreements. A **Contract** is an agreement between two or more competent parties in which an offer is made and accepted, and each party benefits. A contract defines the duties, rights and obligations of the parties, remedy clauses as well as other clauses that are important to characterize the goal of the contract. An **Agreement** is an arrangement between parties regarding a method of action. The goal of this arrangement is to regulate the cooperation actions among partners, and it is always associated to a contract. Examples of agreement clauses are 1) forms of communication, 2) reporting procedures, 3) data representations, etc. Another type of agreement, the **Partnership Agreement,** establishes the provisions that regulate a long-term partnership co-operation. The following cases illustrate some typical organizational forms for cooperative consortia.

Case 1: Explicit consortium (Fig. 4)

– Cooperation is regulated by a contract and a consortium agreement.
– All partners become committed to the Client because they all sign the contract.
– The agreement can be established either before the contract or at contract time.
– The Client cares about who is part of the consortium.
– There are co-operation relationships among partners.
– Apart the commitments represented by the contract and agreement, partners are autonomous.
– The consortium is dissolved at the end of the contract.

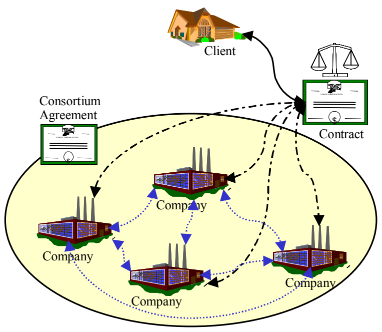


Fig. 4. Explicit consortium

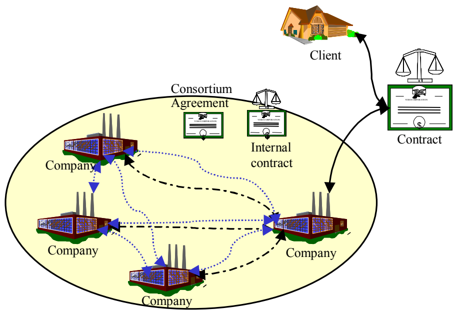


Fig. 5. Internal consortium

Case 2: Internal consortium (Fig. 5)

– There is a contract between one representative of the consortium and the Client.
– The Client doesn't necessarily know about the way the consortium is organized.
– The consortium is also formalized using an agreement and an internal contract.
– Only one partner (the one that signs the contract) is committed to the Client. The other partners are committed to the one that signs the contract.
– There are co-operation relationships among partners.
– Apart the commitments implied by the contract and agreement, partners are autonomous.
– The consortium is dissolved at the end of the contract.

Case 3: Sub-contracting (Fig. 6)

– There is a contract between one partner and a client and subcontracts between this partner and the other service / product providers.
– The Client doesn't necessarily know about the way the contracted partner is organized.

– Only the contracted company is committed to the Client.
– The contracted company establishes all the subcontracts that are needed to perform its contract.
– There is no need for co-operation among the subcontractor partners.
– Apart the commitments represented by the contract/subcontract, partners are autonomous.
– A long-term cooperation relationship may exist, but it is not mandatory.

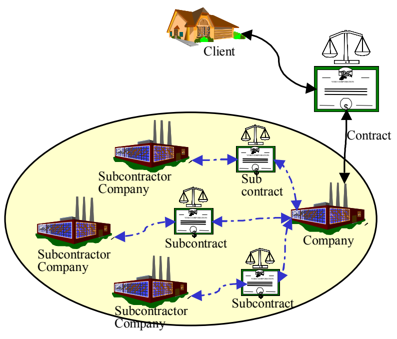


Fig. 6. Sub-contracting

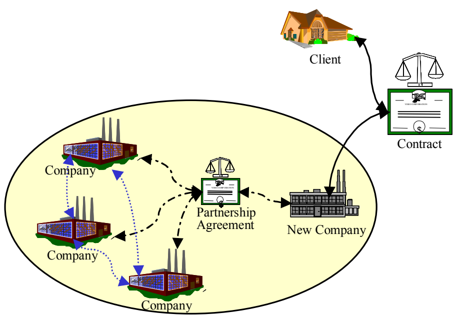


Fig. 7. Partnership

Case 4: Partnership (Fig. 7)
– The partnership creates an entity (new company) using a partnership agreement
– The new company establishes a contract with the Client.
– Only the new company is committed to the Client.
– There are co-operation relationships among partners.
– The partnership may continue after the end of the initial client contract.

Although there have been various works on the organizational issues in MAS, more realistic approaches are necessary in order to adequately model real-world cooperative scenarios as the ones described above, namely in terms of coordination regulated by (declarative) contracts and cooperative agreements. These contracts / agreements constitute the basis for the "social laws" to be followed by the MAS community.

3.4 Distributed Business Process Modeling and Planning

One important aspect in the VE creation is the specification of the tasks necessary to achieve a VE business goal. The activities carried out by a company are usually organized in groups of inter-related activities called *processes* (*business processes*) that can be seen as a set of activities, rules and constraints specifying the steps that must be taken, and conditions that must be satisfied, in order to accomplish a given goal. The composition of each process is designed in order to achieve a (partial) specific goal. A business process can be decomposed into a hierarchy of sub-business processes and enterprise activities. The enterprise activities represent the lowest level in this decomposition, i.e. the basic building blocks that the enterprise must actually realize to perform its business processes.

When a Business Process (BP) is executed by a virtual enterprise, parts of the decomposition of this BP (i.e. sub-processes) are assigned to different enterprises, what makes the BP a ***Distributed Business Process*** (DBP) [20] or a virtual business process in the WISE terminology [4]. When properly "orchestrated", a combination of various processes taking place at different members of the VE will lead to the achievement of the global goal of the VE. The problem of the supervision or coordination of a DBP at its various levels of decomposition is quite important in this context where its definition and enactment is not limited to a single organization, but instead to a set of autonomous, distributed, and heterogeneous nodes that need to cooperate.

The VE coordinator is responsible for the coordination of the entire DBP execution, while the VE members are themselves only responsible for the coordination of the sub-BPs assigned to them. The VE coordinator may itself also act as a VE member, and as such, it can also be responsible for a particular sub-BP. Furthermore a VE member may become the coordinator of a sub-VE inside the VE to coordinate (or supervise) its sub-BP when parts of this sub-BP are assigned to different partners. Under this approach, temporary (sub-) consortia can be formed inside a VE and so on. These sub-consortia are formed for the sole purpose of facilitating the coordination of activities involved in the related sub-business processes. Once a sub-business process ends, the corresponding sub-consortium "dissolves" and its members may become involved in other sub-consortia dynamically formed in this VE as the execution of the VE DBP evolves.

One important question is to determine who is responsible for the creation of BP models and instances. Different VE organizations may consider different actors in this process and different coordination rules [12], [14]. Some possibilities are:

- ***Centralized planning***. In a tightly integrated operation of the VE, the VE coordinator may plan the whole BP and send it to the VE members. In this case it is necessary to take into account the visibility rights of the VE members. Should a VE member see the whole plan or just the part it is responsible for?
- ***Adaptable planning.*** For instance, an intelligent mobile agent can carry a macro plan (abstract BP definition) and detail it once it arrives at a specific VE member according to the specific conditions it finds there. One work in this direction, although applied to remote supervision / tele-operation, can be found in [15], [35].
- ***Cooperative planning***. Another alternative is to consider a cooperative BP planning, by several VE members. In this case it is necessary to implement a shared planning space for the BP model design and negotiation mechanisms for conflict resolution.
- ***Hierarchical planning.*** Finally, the typical case is the one in which only the abstract model of the BP, i.e. only the first few decomposition levels, is planned by the main partner. The main partner is usually the enterprise who identifies the business opportunity, i.e. the broker or VE coordinator. The level of details of this model is just enough to allow the identification of the necessary partners / skills and main resources and the distribution of sub-BPs among these partners [4]. Each VE member is then responsible to refine the assigned sub-BP according to its local capabilities. There might be a need for negotiation in case of conflicts.

A critical aspect at this stage is the ability to share and interlink BP models. Several languages and formalisms have been proposed for BP modeling and coordination. In the VE area many projects adopted a workflow-based approach (**WPDL** – Workflow Process Definition Language) due to both availability of experience with workflow systems in many enterprises and the standardization efforts promoted by the Workflow Management Coalition [36]. This is the case of PRODNET II [11], which developed a modeling tool supporting the following functionality:

- Sequences of activities that might invoke services or other sub-activities.
- Sub-workflow definition, supporting hierarchical (nested) BP modeling.
- Data flow management for parameter passing when activating services / sub-activities, i.e. data that is essential for the process execution control flow.
- Splits and joins that can have the logical conditions AND / XOR.
- Simple and conditional transitions.
- Temporized and cyclic activities.
- Flexible configuration of catalogs of services and relevant data.
- Workflow instances and memory spaces. For each execution of a workflow model an instance is created with its memory space. The explicit data flow associated to an instance (relevant data) is only valid inside the memory space of that instance.
- Management of waiting lists. Each time an instance of a workflow model needs to wait for the conclusion of an external service it is put in a waiting list. Waiting lists are also used for instances waiting for temporized activities. Signals can be sent to the waiting lists manager to provoke changes in the status of workflow model instances.

Figure 8 illustrates the graphical interface for BP (workflow) modeling.

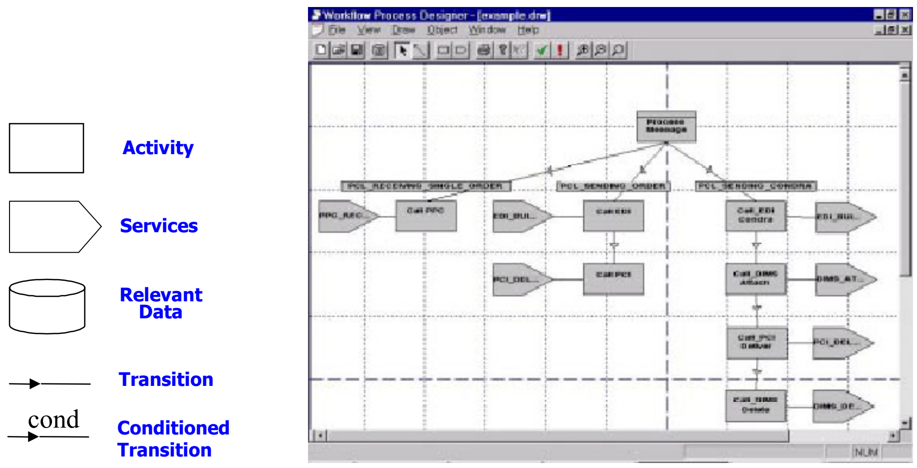


Fig. 8. Graphical process modeling primitives and a BP model editor

Another potential candidate is **PIF** (Process Interchange Format) that emerged in the area of Business Process Reengineering [21] and aims to be a neutral format to act as a bridge across different process representations. A PIF process description consists of a set of objects, such as ACTIVITY, ACTOR, and RESOURCE objects. A design goal for PIF was that its constructs should be able to express the constructs of some existing common process representations such as IDEF0 (SADT) or Petri Nets.

Another example of a process modeling language designed to support interoperability comes from the manufacturing area. Many manufacturing engineering applications use process models, including manufacturing simulation, production scheduling, manufacturing process planning, business process reengineering, product realization process modeling, and project management. Each of these applications embeds different views of processes and use different representations of process information as well. One of the main difficulties with developing a standard to exchange process information is that these applications frequently associate different meanings with the terms representing the information that they are exchanging.

The Process Specification Language (PSL) [31] project of the American National Institute of Standards and Technology (NIST) addressed this issue by proposing a neutral, standard language for process specification to help the integration of multiple process-related applications throughout the manufacturing life cycle. There are four primitive classes, two primitive functions, and three primitive relations in the ontology of core PSL. The classes are OBJECT, ACTIVITY, ACTIVITY_OCCURRENCE and TIMEPOINT. The three relations are PARTICIPATES-IN, BEFORE, and OCCURRENCE-OF. The two functions are BEGINOF, and ENDOF.

There have also been some initiatives towards the merging of PSL with PIF. This combined effort is expected to bring together the representation of both business and manufacturing process-related concepts into a single, unified process modeling language.

The integration of a standard process specification language with an Agent Communication Language is a necessary further step.

Once a global business process is defined, scheduled and responsibilities are assigned to each individual partner, the successful achievement of the common goal – delivery of the final product or service to the client – depends on the proper and timely operation of each VE member (and each supporting service in each VE member). A delay, failure, or even an anticipation of a failure in one node, if not properly attended in time, may jeopardize the common VE goal. Therefore, it is necessary to properly manage (supervise) the inter-dependencies among various (distributed) BPs.

Furthermore, in this domain, the issue of safety is of paramount importance. Therefore, there is a need to integrate in ACL mechanisms for safe communications (cryptography, digital signature, certification, etc.) that have been developed for virtual enterprises and e-commerce.

4. Advanced Information Management and Remote Supervision

4.1 MAS and the VE Operation

Early MAS applications to VE are mainly focused on the creation phase. In many cases it is assumed that simple mechanisms of inter-agent cooperation are sufficient to support the operation phase of VE. With deeper studies of VE application domains however this paradigm reveals many specific aspects that cannot be simply supported by basic MAS approaches.

In the VE community interoperation / cooperation must be regulated by the following features:

- Cooperation agreements and contracts that establish a framework for the general operating conditions must be established.
- Distributed business process models and mechanisms that establish the allocation and sequence of tasks to be performed by the community must exist.
- Efficient data exchange and communication services, distributed service management functionalities, support for nodes autonomy / privacy, high level of service quality, auditability, and accountability, etc., have to be guaranteed.

The decision-making is a hybrid process where it is important to combine human decision with some automatic functionalities. It is even likely that the level of automatic decision-making will evolve as the trust of humans in the systems increases. But independently of the ultimate decision making center, there is a need to provide mechanisms to support process coordination, supervision, and controlled information exchange and sharing.

Some examples of projects that addressed the development of VE operation support mechanisms under a MAS approach are:

– MIAMI [7] that developed a mobile agents platform for VEs supporting a virtual marketplace for VE creation and a monitoring service used during the operation of the VE to supervise processes and provide to partners that have the appropriate permissions with information about the state of partial processes. Global coordination is supported by a workflow management system. Asymmetric cryptography and digital signatures [12] are used to guarantee secure mobility of data.
– MASSYVE [26], [28], already mentioned in the previous chapter, also supports the negotiation process during VE creation within the scope of industry clusters, dynamic scheduling of activities, distributed business process monitoring, and VE reconfiguration. The main focus of MASSYVE is the application of MAS to agile scheduling in VE. The agent nodes represent either enterprises, when the scheduling problem is discussed at the VE level, or the internal manufacturing resources of the company when dealing with internal scheduling of tasks assigned within the company. The Contract-net Protocol coordination mechanism is used to support the task assignment among agents, and the Negotiation method is used to overcome conflicts taking place during planning or execution phases, both at intra-enterprise and inter-enterprise levels.
– MetaMorph II [32] developed a mediator-based multi-agent architecture to support enterprise integration and supply chain management. A manufacturing system is seen as a collection of subsystems, that can be multi-agent based as well, connected through special agents, the mediators. For instance, each enterprise has at least one mediator, representing the administrative center of the enterprise. In the supply chain network, partners, suppliers, and customers are connected through their mediators. But other levels of mediators can exist inside an enterprise. For instance, a tool mediator is used to coordinate all tools that, on their turn, are represented by agents.

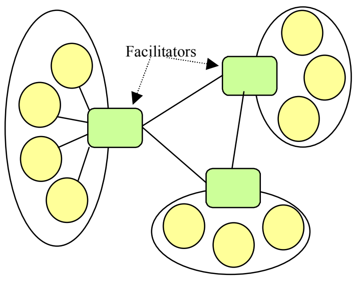
4.2 Federated Approaches

The concept of federation has been emerging in diverse areas such as in the MAS, the database, or the service providing communities. In all these contexts a network of distributed, autonomous, and possibly heterogeneous resources (information or service providers) is considered and the basic principle is to allow a transparent access to these (remote) resources without the need for the client to care about the distribution and communication mechanisms.

Federated MAS. Various federated architectures for MAS have been proposed in the literature [6], [32]. One example is the Facilitator-based approach in which several related agents are combined into a group. Communication between agents takes place always through a specialized interface agent called Facilitator (Fig. 9.a). The main function of the facilitator is to provide communications between a local collection of agents and remote agents through:

- routing messages to the appropriate destinations (via other facilitators), and
- translating incoming messages for the local agents.

Local agents use a restricted subset of an ACL to inform Facilitators about their needs and offerings. Facilitators use this information as well as their knowledge of the global MAS network to transform local agents' messages and route them to other facilitators. Local agents give up part of their autonomy to facilitators and in turn the facilitators satisfy their requirements. An example of this approach can be found in [32].



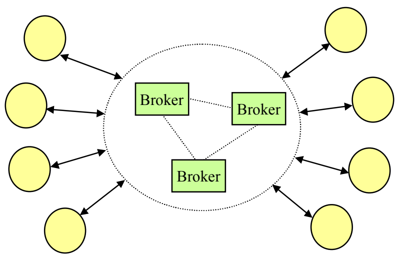


Fig. 9. a) Facilitator-based federation b) Broker-based federation

Another case is the broker-based federation (Fig. 9.b). Brokers are agents similar to facilitators but with some additional functions such as monitoring and notification. While a facilitator is responsible only for a designated group of agents, any agent may contact any broker in the same system for finding service / information agents for a particular task.

Federated databases. A federated database system is a distributed multi-database system in which every node in the federation maintains its autonomy on the data and defines a set of export schemas through which the data is made available to other specific nodes. Every node is able to import data from other nodes through their import schemas, and access their data according to the bilaterally pre-defined access permissions. As a consequence of this general interaction facility, the approach allows the cooperation between federated nodes, in order to accomplish a common global task, while the autonomy and independence of every node is preserved and reinforced. Several schemas represent every node as represented in Fig. 10 [1]:

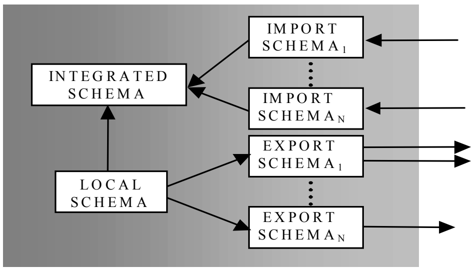


Fig. 10. Schemas representation in one federated database node

- A local schema: the schema that models the data stored locally.
- Several import schemas: the various import schemas model the information that is accessible from other nodes.
- Several export schemas: an export schema models some information that the node wishes to make accessible to other nodes (usually, a node defines several export schemas).
- An integrated schema: which presents a coherent view on all accessible local and remote information.

Due to its capability to preserve node's autonomy while supporting cooperation and transparent data access via its federated query processing mechanism, the federated distributed architecture is a strong base approach for information management in VE. The federated query-processing element is responsible for provision of access to the information for which an enterprise is authorized, while preserving all nodes' autonomy, visibility levels, and access rights for exchanged information among VE nodes. PRODNET II [8], [12], [17] is an example project that adopted a federated distributed approach for its VE infrastructure.

Federated databases and MAS. In the MASSYVE project [26], [28] an integration of MAS and federated information management is proposed. Each agent is enhanced with a Federated Information Management System (called FIMS), through which it seamlessly interoperates and exchanges information with other agents. However, considering the autonomy of agents, the access to information is strongly controlled by the information visibility rights defined among them that in turn preserve their autonomy. Therefore, a MASSYVE Agent is seen as a kind of tandem architecture composed of a "normal" agent and its FIMS. Fig. 11 illustrates the architecture of a MASSYVE agent.

An essential concept introduced in this architecture is that the *data* are not sent from one agent to the other via a high-level protocol (e.g. ACL language), as in the traditional *push* strategy case, but rather through the *pull* strategy, via accessing to the respective agents' FIMSs. Thus, the high-level protocol is only used for the control/coordination purposes.

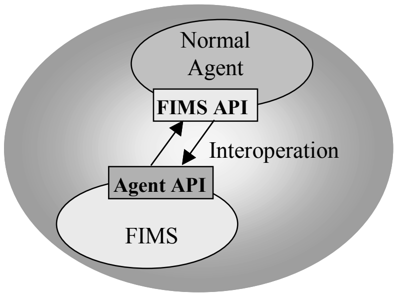


Fig. 11. Structure of a MASSYVE Agent

Figure 12 illustrates the MASSYVE approach for data exchange. Consider an example case where a given agent (B) processes some information and generates some results (for example the “actual end of the production date for a part P”) that are needed to be accessed by another agent (A), according to the predefined *supervision clauses* specified in the VE’s contract. Following the contract, then B sends a message to A (represented by “1” in this Figure), communicating that the data item on “part P’s actual end of the production date” is at this enterprise. This control message sent from B to A informs A that now this data item is available and can be accessed by A (through its FIMS’ *import schema*). Please notice that the access rights for the shared data among nodes are dynamically and bilaterally configured and preserved by their *import/export schemas* according to agents’ roles in the collaboration and their needs.

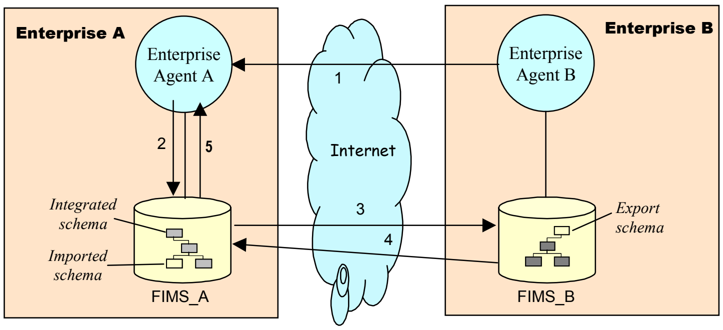


Fig. 12. Exchange of data in a *pull* strategy.

Once this message is received at A, (represented by “1”), whenever A wishes it can retrieve this updated information from B. In fact, this access goes as follows: Agent_A queries this information from its own integrated schema in FIMS (represented by “2”) requesting for the actual end of production date on part P; an automatic access will occur from FIMS_A to FIMS_B (represented by “3”) using the federated mechanisms for information access – that receives and returns this authorized information from B (represented by “4”). This information is in turn returned from FIMS_A to Agent_A for its internal processing (represented by “5”).

Furthermore in the general case, the federated query mechanism of FIMSs allows agents, for instance Agent_A, to query information from their integrated schema that might initiate several queries to different other enterprises, transparently collecting partial contributions that are finally assembled into a global answer from the FIMS to the agent. The actual information exchange among VE nodes may resort to a safe and secure communications layer as the PCI module developed in PRODNET II [12].

The main advantages of this approach follow:

- The inter-agent message's content becomes shorter and leaner;
- Agents will always access the necessary up-to-date data from their sources, at the exact time the data are needed;
- Transparent and controlled access to distributed data is provided over the agents' network in an integrated method. In this way, the agents can concentrate their tasks on the reasoning and processing of information instead of the management of information;
- Data and control is totally separated from each other in the multi-agent interaction environment;
- The information access rights and visibility levels among agents can be defined efficiently and evolve dynamically using the federated information management system functionalities;
- VE agents can only access authorized data with respect to their current access right definitions so that agents' desirable autonomy is preserved in terms of their data.

Services federation. In the service federation approaches, like the one illustrated by the Jini-based architecture [34] being applied in the FETISH project [2], service providers that, independently of the way their services are implemented or located, make them accessible in a kind of "virtual market". A client such as a VE creator can "shop" in this market for the best set of services to satisfy the needs of a given business opportunity. The service federation infrastructure provides the basic mechanisms for transparent (remote) access to services according to some agreed access rules.

Services are registered in Service Catalogs and various catalogs may be interconnected. Advanced lookup services will support service discovery and selection. Due to the members autonomy (and legacy) there might be a large heterogeneity / diversity in the way services are implemented. However, in order to facilitate service selection ("shopping") and utilization, a common service interface needs to be agreed among the service providers and in case of legacy implementations a service adapter has to be developed. This interface can be decomposed in two parts: service specification descriptor and service invocation wrapper (or proxy), i.e. the service API. The service specification describes the characteristics of the service such as service identifier, functionality, I/O parameters, applicability conditions, access rights, etc. The service invocation wrapper is a software component that provides a transparent way to invoke the service, hiding the details of the physical distribution and implementation.

It could be interesting to investigate the merging of the MAS federation and Service federation approaches. In particular, some of the advanced look up mechanisms developed for service federation could be used in the implementation of the virtual market mechanism of a federated multi-agent system (extending the initial yellow pages approaches).

4.3 Cooperation Shared Spaces

An example of a very demanding application in manufacturing is the Collaborative or distributed Concurrent Engineering. Product development needs the interdisciplinary contribution of design, process and manufacturing engineers, and other contributors with diversified expertise, each one contributing with a different skill. The efforts to shorten this development phase led to the overlap of activities and to the introduction of mechanisms to coordinate them. The concept of Concurrent Engineering (CE) is the result of the recognition of this need to integrate diversified expertise and to improve the flow of information among all phases and actors involved in the product life cycle.

Various projects have addressed the development of cooperative platforms to support CE and covering four areas: interoperable computational and communication infrastructures, common information models / ontologies, engineering information management, and process execution and supervision (coordination). In the context of a VE these teams will involve experts located in different geographical locations and belonging to different enterprises, what makes the need for an adequate support infrastructure more acute.

Various tools from the CSCW area can be useful in this domain but more specific tools are necessary. As the participants are not located in the same place and eventually the work is developed with different time schedules (asynchronous processes), it is very important to support:

(i) Sharing of information models and process models, describing the product model and its manufacturing process and the design/planning process itself. The requirement is not only for a bi-lateral exchange of information, but also to the establishment of shared spaces.

(ii) Provision of adequate visibility and access rights definition and management.

(iii) Coordination of (asynchronous) activities performed in different places by different actors.

(iv) Provision of notification mechanisms regarding major events in the design / planning process (e.g. conclusion of a step by one actor).

The federated database paradigm represents a suitable approach to develop shared spaces with the appropriate mechanisms to specify and ensure the visibility levels and access rights as represented in the CIMIS.net project [3]. A flexible notification mechanism can also be implemented by combining the federated information management with a workflow-based coordination system [8]. But additional coordination mechanisms can be supported by the use of a multi-agent approach.

For instance in the CIM-FACE system [24], [25], the human experts use the enterprise Computer-Aided tools (e.g. CAD, CAPP, CAM, etc.) to perform their design and planning tasks. In order to facilitate the coordination among team members, the concept of process assistant agent is introduced (Fig. 13). One process assistant is created to assist each human member of the Concurrent Engineering team. This assistant is an agent that represents the human expert in the virtual sharing community and provides the human with a “window” to the process model he/she is involved in. The coordination of activities among the team members is ensured by these special agents that are responsible for:

- Keeping track of the evolution of the collective design / planning process.
- Notifying other members of the team (via the corresponding assistant agents) of important events.
- Checking and ensuring the required precedence rules in the flow of activities.

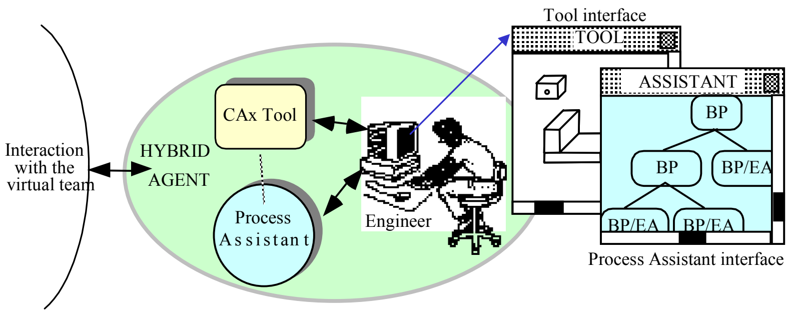


Fig. 13. Process assistant agent in Concurrent Engineering

In its current implementation, the CIM-FACE prototype uses a limited centralized information management system. A promising approach requiring further investigation is the combination of the federated information management paradigm with the multi-agent assistant community.

4.4 Mobile Agents Supporting Remote Supervision

Once a basic infrastructure is established, new forms of advanced cooperation among VE members and the need for proper functional support will naturally emerge. In particular, advanced forms of cooperation mostly in the area of design and manufacturing require mechanisms to support a controlled “intrusion” of a company, for instance the VE coordinator, into the “territory” of its partners. An initial example of this “intrusion”, which is properly supported by the federated database paradigm, is the access to selected (authorized by the cooperation agreements) subsets of the information (for instance, the orders’ status, stock levels, etc.) [20]. But this process may assume more extensive forms. Consider the case that a company wishes to “open a window” over the shop-floor of its partner to monitor the manufacturing process of the ordered parts and even have an interference on, i.e. supervise from distance and in cooperation with the local people, the shop-floor processes.

Supervision represents a collection of inter-related activities including task dispatch and execution, execution monitoring, error diagnosis and recovery. When considered in a distributed environment like in a VE, the concept of ***remote supervision*** emerges. If the supervision process involves the collaboration of various actors, located at different remote places, we have ***collaborative remote supervision***.

The design of a proper support system for collaborative remote supervision (CRS) can benefit from the contributions coming from a number of areas that, although conceptually close, are usually addressed by different communities of researchers with little interaction among them. The two main contributing areas to remote supervision are the Telerobotics and Virtual Laboratories. Furthermore, other areas of research and development contributing to CRS include the Virtual Reality, Virtual Organizations, and Computer Supported Cooperative Work.

The remote operation of machinery has been addressed for many years, mainly for security related applications. However, the Internet has been opening new opportunities for remote operation due to low costs and widespread availability, what makes it very appealing as a basis for remote operation. In fact several examples of connection of robots, cameras, and other devices to the WEB were implemented during the last years. Remote operation in manufacturing via Internet suffers however, from several problems: i) Internet is characterized by long and irregular time-delays and very often, suffers from low levels of availability, raising new challenges in what concerns the reliability of the implemented system and its dependence on the characteristics of the network; ii) when reasonable practical application domains are considered, high levels of heterogeneity are expected in the availability of sensors and equipment at the remote places, which can degrade the flexibility and scalability of the system; and iii) the composition of the execution environments are potentially unstructured and unknown, which means that it is not adequate to resort to deterministically programmed systems. Complementarily, the increased use of wireless networking (mobile / ubiquitous computing) requires short connection periods.

In order to cope with the mentioned difficulties, an approach based on adaptive mobile agents was developed by the Robotics and CIM group at the New University of Lisbon [15], [35]. The mobile agents paradigm shows important advantages when remote manipulation and remote supervision are considered, since: i) moving the code to the places where the machines and sensors are located, contributes to enable close to real-time response, and so, the availability, delays and reliability of the network become less of a problem; ii) new mobile agents can be built and sent for remote execution whenever needed, thus greater flexibility and scalability is achieved. Furthermore, the use of mobile agents has other potential benefits: i) it eases the correction of errors in software; ii) it eases rescue operations if an autonomous vehicle gets lost; iii) it, naturally, leads to adaptive and up-to-date systems; and iv) it eases remote maintenance and diagnosis.

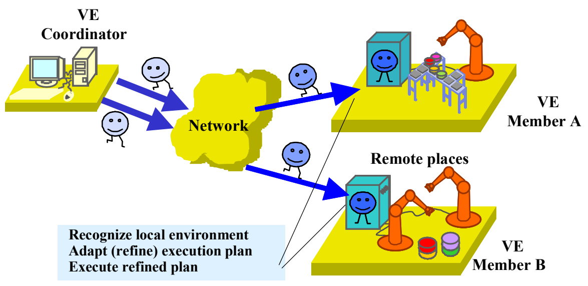


Fig. 14. Mobile agents in a VE supervision environment

However, in order for the same mobile agent to be executed at several places, it must carry only a general action plan, which must be refined/adapted when the agent reaches every target place. Therefore, agents must be equipped with decision-making and plan refining capabilities, which allow them to, based on the abstract (hierarchical) plans they carry, build specific plans suitable for execution in the particular environment of each site they reach (Fig. 14).

In addition to the mentioned generic advantages of the mobile agents approach [15], the remote supervision application can also benefit from the autonomy of the agents in order to not require a synchronous availability of the participants in different nodes as these participants can delegate to their agent representatives the actual realization of some task, which will be done when the necessary conditions are satisfied.

When dealing with remote operation, the agents must run with a high degree of autonomy in uncertain environments. To achieve this goal, an approach based on general monitoring and recovery methods and on plans with annotations, intended to help the execution monitoring and error recovery, was adopted in the mentioned work. A hierarchical plan structure was considered, since, besides other advantages of hierarchical planning, it allows the specification of monitors at various levels of detail, which is quite appropriate for complex domains. Furthermore, the hierarchical approach is a powerful mean to structure interesting monitoring strategies that range over a set of low level actions.

5. Conclusions and Open Challenges

The virtual enterprises / virtual organizations areas represent a fast evolving research domain and a promising application field for multi-agent approaches. In fact several recent projects have addressed the application of the MAS paradigm and related market-based negotiation mechanisms for the partners selection and task allocation during VE creation. Also some attempts to develop MAS-based infrastructures to support the operational phase of the VE have been made. Nevertheless several challenges remain open for MAS requiring further research, such as:

- Support for the full life cycle of the VE.
- Adoption of contract-based coordination models.
- Necessary integration of MAS with several other paradigms.
- Interoperation with legacy systems and enterprise applications.
- Inclusion of specialized protocols and standards.
- Support of robust safety mechanisms.

The emergence of new organizational forms and new cooperation paradigms provides an appropriate ground for the development and validation of advanced MAS organizations.

Finally it shall be noticed that in order to be accepted by the industrial community, MAS applications need to be successfully demonstrated in complex real world pilot systems.

Acknowledgements. This work was funded in part by the European Commission through the Esprit PRODNET II, INCO MASSYVE, and IST FETISH-ETF projects.

References

1. Afsarmanesh, H.; Tuijnman, F.; Wiedijk, M.; Hertzberger, O. – Distributed schema management in a cooperation network of autonomous agents, Proceedings of DEXA'93, Prague, Czech Republic, 1993.

2. Afsarmanesh, H.; Camarinha-Matos, L.M. - Future smart organizations: A virtual tourism enterprise, *Proceedings of WISE 2000 – 1st ACM/IEEE International Conference on Web Information Systems Engineering*, Vol. 1 (Main Program), pp 456-461, IEEE Computer Society Press, ISBN 0-7695-0577-5, Hong Kong, 19-20 June 2000.
3. Afsarmanesh, H.; Wiedijk, W.; Ferreira, A.C.; Moreira, N.P. - Distributed Database Support for a Concurrent Engineering Environment. *Journal of Studies in Informatics and Control, Vol. 3, Nos. 2-3*, pages 135-143. IC Publications, Romania, Sept 1994.
4. Alonso, G.; Lascano, A.; Schuldt, H.; Schuler, C. - *The WISE approach to Electronic Commerce*, http://www.inf.ethz.ch/department/IS/iks/research/wise.html, Feb 15, 1999.
5. Azevedo, A.; Sousa, J.; Bastos, J.; Toscano, C. – A distributed order promise and planning system for the virtual enterprise, in Globalization of Manufacturing in the digital communications era of the 21st century, Kluwer Academic Publishers, ISBN 0-412-83540-1, 1998.
6. Beasley, M.; Cameron, J.; Girling, G.; Hoffner, Y.; Linden, R.; Thomas, G. – Establishing co-operation in federated systems, Systems Journal, Vol. 9, Iss. 2, Nov. 1994.
7. Broos, R.; Dillenseger, B.; Guther, A.; Leith, M. – MIAMI: Mobile intelligent agents for managing the information infrastructure, www.infowin.org/ ACTS/ANALYSYS/PRODUCTS/THEMATIC/AGENTS/ch3/miami.htm, 2000.
8. Camarinha-Matos, L.M; Afsarmanesh, H.; Garita, C.; Lima, C. - Towards an architecture for virtual enterprises, *Journal of Intelligent Manufacturing*, Vol. 9, Issue 2, Apr 1998.
9. Camarinha-Matos, L. M.; Afsarmanesh, H. - Infrastructures for Virtual Enterprises - Networking Industrial Enterprises, Kluwer Academic Publishers, ISBN 0-7923-8639-6, Oct 1999.
10. Camarinha-Matos, L. M.; Cardoso, T. – Selection of partners for a virtual enterprise, in [9], 1999.
11. Camarinha-Matos, L. M.; Carelli, R.; Pellicer, J.; Martin, M. - Towards the virtual enterprise in food industry, L.M. Camarinha-Matos, in Re-Engineering for Sustainable Industrial Production, Chapman & Hall, ISBN 0-412-79950-2, May 1997.
12. Camarinha-Matos, L. M.; Afsarmanesh, H.; Osorio, A.L. - Flexibility and safety in a web-base infrastructure for virtual enterprises, L.M. Camarinha-Matos, H. Afsarmanesh, A. L. Osório, *International Journal of Computer Integrated Manufacturing* (Taylor & Francis), Vol. 14, N. 1, Jan 2001.
13. Camarinha-Matos, L.M.; Pantoja Lima, C. - A Framework for Cooperation in Virtual Enterprises, Proceedings of DIISM'98 - Design of Information Infrastructures Systems for Manufacturing, Fort Worth, USA, May 1998.
14. Camarinha-Matos, L.M.; Lima, C. - Supporting business process management and coordination in a virtual enterprise, *in Advances in Networked Enterprises, Kluwer Academic Publishers,* ISBN 0-7923-7958-6, Sept 2000.
15. Camarinha-Matos, L.M.; Vieira, W. - Intelligent mobile agents in elderly care, *Journal of Robotics and Autonomous Systems* (Elsevier), Vol. 27, N. 1-2, ISSN 0921-8890, Apr 1999.
16. Davidrajuh, R.; Deng, Z. Q. – Identifying potential supplier for formation of virtual manufacturing systems, Proceedings of 16th IFIP World Computer Congress 2000, Vol. ITBM, Beijing, China, 21-25 Aug 2000.
17. Garita, C.; Ugur, Y.; Frenkel, A.; Afsarmanesh, H.; Hertzberger, L.O. - DIMS: Implementation of a Federated Information Management System for PRODNET II, Proceedings of 11th International Conference and Workshop on Database and Expert Systems Applications - DEXA '2000, London, England, 2000.

18. Goranson, H.T.– The Agile Virtual Enterprise – Cases, metrics, tools. Quorum Books, ISBN 1-56720-264-0, 1999.
19. Kaihara, T. – Supply chain management based on market mechanism in virtual enterprise, in [9], 1999.
20. Klen, A.; Rabelo, R.; Spinosa, M.; Ferreira, A. – Distributed business process management, in [9], 1999.
21. Lee, J.; Yost, G. - The PIF Process Interchange Format and Framework, Version 1.0, http://ccs.mit.edu/pifmain.html, Dec22, 1994.
22. Li, Y.; Huang, B.Q.; Liu, W. H.; Wu, C.; Gou, H.M. – Multi-agent system for partner selection of virtual enterprises, Proceedings of 16th IFIP World Computer Congress 2000, Vol. ITBM, Beijing, China, 21-25 Aug 2000.
23. OAG – Open Applications Integration White Paper, Open applications Group, 1997.
24. Osorio, A.; Camarinha-Matos, L.M. – A federated multi-agent infrastructure for concurrent engineering (CIM-FACE), Studies in Informatics and Control, Vol. 5, N. 2, June 1996.
25. Osorio, A.L.; Oliveira, N.; Camarinha-Matos, L.M. - Concurrent Engineering in Virtual Enterprises: The extended CIM-FACE architecture*, Proc. Of BASYS'98 – 3rd IEEE/IFIP Int. Conf. On Balanced Automation Systems, Intelligent Systems for Manufacturing* (Kluwer Academic), ISBN 0-412-84670-5, Prague, Czech Republic, Aug 1998.
26. Rabelo, R.; Afsarmanesh, H.; Camarinha-Matos, L.M. - Federated multi-agent scheduling in virtual enterprises, in *E-business and Virtual Enterprises, Kluwer Academic Publishers,* ISBN 0-7923-7205-0, Oct *2000.*
27. Rabelo, R.; Camarinha-Matos, L.M. - Negotiation in Multi-Agent based dynamic scheduling, *Int. Journal on Robotics and CIM,* Vol. 11, N. 4, Dec 1994.
28. Rabelo, R.; Camarinha-Matos, L.M.; Afsarmanesh, H. - Multi-agent-based agile scheduling, *Journal of Robotics and Autonomous Systems* (Elsevier), Vol. 27, N. 1-2, ISSN 0921-8890, Apr 1999.
29. Rabelo, R.; Camarinha-Matos, L.M.; Vallejos, R. - Agent-based brokerage for virtual enterprise creation in the moulds industry, *in* E-business and Virtual Enterprises, Kluwer Academic Publishers, ISBN 0-7923-7205-0, pp.281-290, Oct 2000.
30. Rocha, A.; Oliveira, E. – An electronic market architecture for the formation of virtual enterprises, in [9], 1999.
31. Schlenoff, C.; Gruninger, M.; Tissot, F.; Valois, J.; Lubell, J. Lee, J. - The Process Specification Language (PSL) - Overview and Version 1.0 Specification, NIST Internal Report (NISTIR) 6459, http://www.mel.nist.gov/psl/, 1998.
32. Shen, W. – Agent.based cooperative manufacturing scheduling: an overview, COVE News N. 2, www.uninova.pt/~cove/newsletter.htm, Mar 2001.
33. Shen W. & Norrie, D.H. - An agent-based approach for distributed manufacturing and supply chain management, in *Globalization of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise,* Jacucci, G. (ed.), Kluwer Academic Publishers, 1998.
34. SUN, 1999 - JINI Technology Architectural Overview, http://www.sun.com/jini/whitepapers/architecture.html, Jan 1999.
35. Vieira, W.; Camarinha-Matos, L.M. - Adaptive mobile agents: Enhanced flexibility in Internet-based remote operation, *in Advances in Networked Enterprises, Kluwer Academic Publishers,* ISBN 0-7923-7958-6, Sept 2000.
36. WfMC - Workflow Management Coalition (1994) - The Workflow Reference Model - Document Nr. TC00 - 1003, Issue 1.1, Brussels, Nov 29, 1994.
37. Zarli, A.; Poyet, P. – A framework for distributed information management in the virtual enterprise: The VEGA project, in [9], 1999.

Specialised Agent Applications

Klaus Fischer, Petra Funk, and Christian Ruß

German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany
{Klaus.Fischer|Petra.Funk|Christian.Russ}@dfki.de,
http://www.dfki.de/{∼kuf∼funk∼russ}

Abstract. With the ever growing usage of the world wide IT networks, agent technologies and multiagent systems (MAS) are attracting more and more attention. (Multi-)Agent technologies aim at the design of agents that perform well in environments that are not necessarily well-structured and benevolent. This article tries to give an overview over MAS applications. However, because of the lack of space and time it is not possible to make this overview comprehensive in any sense. We therefore concentrate on the application of MAS in the context of supply chain management in virtual enterprises. Additionally, we give pointers to related work and general literature for application-oriented research of MAS. In MAS applications emergent system behaviour is one of the most interesting phenomena one can investigate. However, there is more to MAS design than the interaction between a number of agents. For an effective system behaviour we need structure and organisation. To achieve this we present the concept of holonic multiagent systems and demonstrated how it can be utilised in the selected application domain.

1 Introduction

From the very beginning multiagent system (MAS) research has been application-oriented. Early work is documented in [15], [11], and [2]. More up-to-date overviews of MAS applications can be found in [20] and [26]. Current activities in Europe can be found under the special interest group activities of AgentLink[1]. While the early work was done in the context of distributed artificial intelligence (DAI), the increasing importance of the world wide telecommunication and computer networks, especially the Internet and the World Wide Web (WWW), brought about increasing interest in agent technologies in the past few years. Although in many of todays applications individual agents are trying to fulfill a task on behalf of an individual user, these agents are doing so in a multiagent context. It is obvious that the problem solving capabilities of MAS will again become more and more important. However, the implementation of MAS for interesting real-world application scenarios tend to be very complex. The basic MAS approach to tackling this complexity is to base problem solving on emerging bottom-up behaviour. This is achieved by giving the agents specific abilities

[1] http://www.AgentLink.org/

which lead to emergent problem solving behaviours when the agents interact with each other. Although this approach works well in many cases, the solutions tend to be sub-optimal. To enhance such solutions, we advocate the concept of holonic MAS which introduces structure in a society of autonomous agents.

We use supply chain management in virtual enterprises as the main application scenario throughout the article. Many of the basic technologies that are investigated in MAS research can be applied in this context and there is significant commercial interest in this topic. According to Booz-Allen & Hamilton, in retail industry as well as in many other industries supply structures are evolving which may be characterized as "...complex sets of relationships that appear more web-like than chain-like." Such structures are called supply webs [18]. Partnerships between autonomous business entities in these supply webs can be flexibly contracted or withdrawn and are predominantly short-dated. This may cause complex coordination problems since the resulting many-to-many interactions and instantiated supply paths are not stable but may dynamically change. The coordination of the manifold interactions and internal planning tasks over strains present-day inventory control systems.

After we introduced the basic concepts of holonic MAS in Section 2, we present in Section 3 a framework for agent-based supply web management. Section 4 presents the design of a production node in a supply web. A MAS design of the control of a manufacturing systems is described. We show how the holonic design of the overall supply chain continues in the internal structure of the MAS for the control of a flexible manufacturing system. Section 5 presents the TeleTruck system a MAS for online dispatching in haulage companies. With the TeleTruck system a haulage company can be included in the overall supply chain frame work in a flexible manner. Finally, in Section 6 we present a MAS for distributed software management.

2 Holonic MAS

We already explained that the basic MAS approach to tackling problem solving in complex domains is to base it on emerging bottom-up behaviour. With the Watchmaker's parable Simon demonstrated that a hierarchy offers another useful paradigm for tackling complexity [23]. The hierarchical solution to a global problem is built up from modules which form stable sub-solutions, allowing one to construct a complex system out of less complex components. Control in such a hierarchy can be designed in a centralised or a decentralised manner. The decentralised model offers robustness and agility with respect to uncertainties in task execution. The major advantages of the introduction of centralised planning and control instances are predictability, opportunities for performance optimisation, and an easier migration path from current to distributed systems [3].

To design and implement systems that include both hierachical organisational structures as well as decentralised control the concepts of *fractal* and *holonic* design were proposed [25, 5]. The word *holon* [17] is derived from the Greek *holos* (whole) and the suffix *on*, which means particle or part. A holon is a natural or artificial structure that is stable, coherent, and consists of several holons as substructures. No natural structure is either *whole* or *part* in an absolute sense. A holon is a complex whole that consists of substructures as well as being a part of a larger entity. In both approaches, in fractal as well as holonic design, we have the ideas of recursively nested self-similar structures which dynamically adapt themselves to achieve the design goals of the system. We adopt the notion of *holonic multiagent systems* to transfer these ideas to the design of MAS. In a holonic MAS autonomous agents group together to form holons. However, in doing so they do not loose their autonomy completely. The agents can leave a holon again and act autonomously or rearrange themselves as new holons. According to this view a holonic agent consists of sub-agents, which can separate and rearrange themselves and which may themselves be holons.

These ideas are certainly not unfamiliar to the multiagent community since Marvin Minsky proposed his vision of the mind as a society of agents in [19] and tried to show how human intelligence could evolve from the neuronal structures of the brain. His approach was to show how complex cognitive tasks can be decomposed into simpler ones that can then be performed by simple, non-intelligent computational structures called agents. Minsky's main message is that the human mind consists of less intelligent structures, which in turn consist of simple parts that are not intelligent at all.

In a MAS holonic structures occur when agents not only cooperate loosely but have to be composed in order to perform their tasks. That means autonomous agents join others and form holons without loosing their autonomy completely but keeping the freedom to leave the holon again and act autonomously or rearrange themselves as new holons. According to this view we call an agent holonic or a *holon* if it consists of subagents, which can rearrange themselves and which may again be holonic. The "leaves" of a holonic MAS are autonomous agents that are stable over time and cannot be decomposed further into subagents[2]. We will assume one subagent of a holon to be distinguished as the representative or *head*, that will represent the holon to the rest of the agent society according to all kinds of interactivity.

The competences of the head of a holon range from pure administrative tasks to the authority to issue directives to the other agents. The minimum task a representative must implement is the interaction with the rest of the agent society, such that the behaviour of the holon is consistent. Furthermore, the head can be equipped with the authority to allocate resources to the other agents in the holon, to plan and negotiate for the holon on the basis of its subagents' plans and goals, or even to remove parts from the holon or to incorporate new parts.

There is no universal method to determine the head. It may be elected from the agents forming the holon; a new agent may be created just for the lifetime of the holon to represent it; or, as in our setting, special agents are designed, that coordinate the formation of holons and establish themselves as heads of these holons.

[2] Nevertheless, it is possible to find holonic structures in the sense of Koestler in these agents' architecture [12].

3 Multiagent Co-ordination in Supply Webs

This section presents a framework for agent-based co-ordination in a supply web. We present the design of a warehouse agent as an example of a supply web agent (SWA) and an architecture for an agent-based co-ordination server.

The supply web application domain can be structured into three different layers each of them representing a different view on the domain (Fig. 1). The first layer represents the *supply web structure*. As in any supply chain available resources are (i) transformed to goods or products along some supply paths, (ii) allocated to brokers, wholesalers and/or retail combines, and finally (iii) distributed by them (potentially via additional intermediators) to the consumers. However, in modern competitive settings of electronic markets supply paths are becoming more and more established quite short-dated, that is not until a real supply flow between appropriate supply web entities is impending.

We tackle these co-ordination problems by applying the concept of intelligent agents to the design of retail supply webs. Each operative unit of a retail supply web is modeled as a SWA which represents either a supplier, producer, broker, wholesaler, retailer, warehouse, distribution center, branch, or logistics service provider. The co-ordination infrastructure for these supply web agents relies on co-ordination agents, services, and mechanisms such as auction server agents providing, e.g., matchmaking services, and auction types, coalition forming mechanisms for (re-)allocation of resources, and distributed business and supply processes.

This implies the need to flexibly exchange parts of supply paths. The situation becomes even more complex in situations where the outcome of the competition of supply web entities for the assignment of goods in the supply chain is not known in advance (see circle in Fig. 1). Depending on who wins this competition all of the subsequent entities in the winning supply path are enforced to adapt their plans correspondingly. The second layer models what we can consider as the *agentification view*, i.e., how a set of supply web agents is mapped to corresponding physical entities within the supply topology. Finally, on top of these layers the *co-ordination layer* consists of two main components: a mediation infrastructure providing agent naming or yellow page services, and a co-ordination infrastructure which consists of special co-ordination agents endowed with appropriate co-ordination mechanisms.

The holonic warehouse agent is an example of a special SWA. This holon consists of different types of interacting agents: The supervisor agent represents the interface to the supply web environment and co-ordinates the operation of its subagents; the auction agent has to handle all activities related to auctions; the stock agent manages the flow of goods as well as policy based reservations.

Co-Ordination Policies: Each warehouse agent in a supply web behaves according to given rules (policies) of which the main ones are the following.

Supervisor Policy: The supervisor agent decides on which good has to be traded at what time and for what price. For example, the keep-the-level

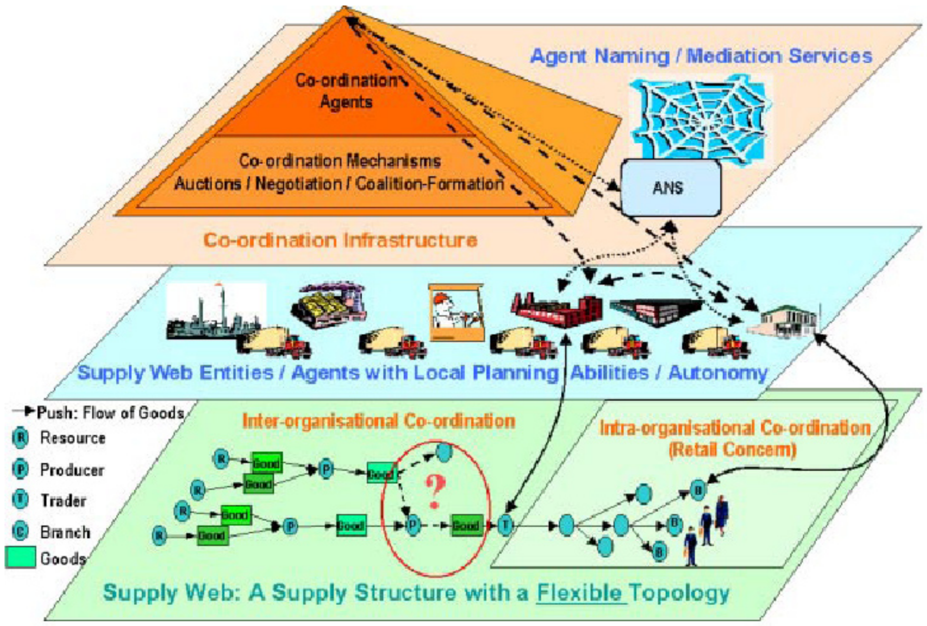

Agent Naming / Mediation Services
Co-ordination Agents
Co-ordination Mechanisms
Auctions / Negotiation / Coalition-Formation
ANS
Co-ordination Infrastructure
Supply Web Entities / Agents with Local Planning Abilities / Autonomy
Push: Flow of Goods
Resource
Producer
Trader
Branch
Goods
Inter-organisational Co-ordination
Intra-organisational Co-ordination (Retail Concern)
Good
?
Supply Web: A Supply Structure with a Flexible Topology

Fig. 1. Three views on the supply web application domain.

policy describes the re-ordering of goods in case an entity falls short of storage capacities.

Bidding Policy: The auction agent determines the value of the bids. For example, the always-one-higher policy prescribes a continuous increase in bid value by one until a certain limit is reached.

Decommitment Policy: Any commitment or reservation of resources may be canceled with penalty. Various policies can manage the decision if and when to decommit.

Reservation Policy: An agent has to decide on if some proposed reservation of goods for another agent is acceptable, or not.

Other behavior-oriented policies include policies for supply paths, and calculation of delivery dates and delays. Latter kind of policies can be categorized into just-in-time policies, safety lead-time policies, bid refusal policies, and promise date negotiation policies. A comprehensive overview of these types of policies can be found in [16]. Interactions between several SWAs take place according to their local policies over a supply co-ordination server that we are going to describe in the next paragraph.

The agents in the supply web communicate and co-ordinate their activities by means of a holonic supply web co-ordination server. This server has been structured into a 3-layered holonic agent societies as follows: The Co-ordination Matchmaker Agent (CMM Agent) serves as an interface to SWAs which are requesting for co-ordination of their sales activities in terms of an auction. For

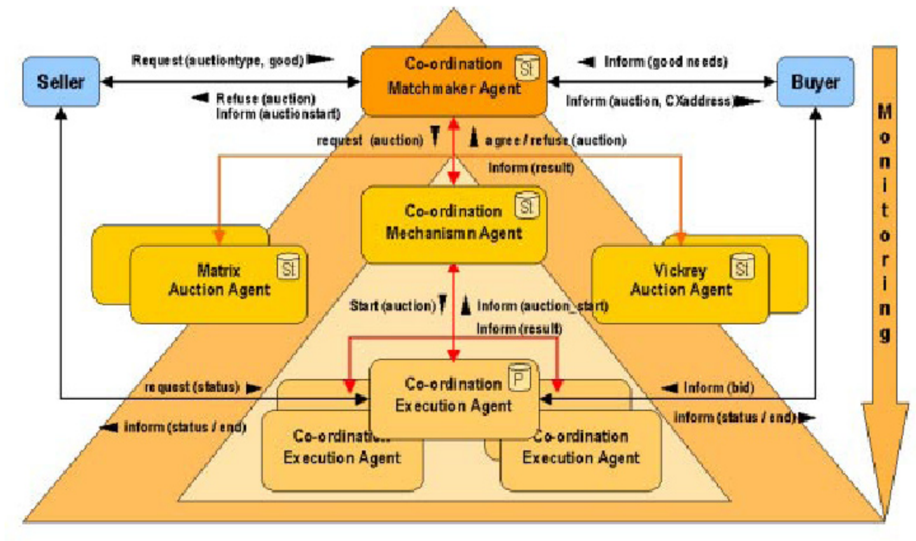


Fig. 2. The supply web co-ordination server.

each request the CMM agent selects the most appropriate co-ordination mechanism (CMech) agents for further processing of the request. In case there are enough resources available for initiating an auction of the desired type by the subordinated co-ordination execution (CX) agents the respective SWA will be notified on the details of this special auction matching its request. If the request matches with some currently running auction(s) the CMM agent forwards the contact details of the corresponding CX agent(s) it receives by the respective CMech agent(s) to the SWA.

The SWA can then directly contact the selected CX agent(s) for bidding or status monitoring depending on its role as buyer or seller, respectively. The CMM as wll as the CMech agents are in charge of efficiently co-ordinating and monitoring operative processes of running auctions to ensure appropriate load balancing at the server. While the CX agents store the data of running auctions, they propagate the result of each auction for reasons of statistics also internally in a bottom-up fashion to each of the responsible agents, i.e., the CX, CMech, and CMM agent(s).

Market-based Supply Web Co-Ordination Mechanisms: We have endowed our supply web co-ordination server with several market-based allocation mechanisms for the co-ordination of supply web activities such as the simulated trading algorithm [1] and matrix auction [14].

The simulated trading (ST) algorithm is a randomized algorithm that realizes a market mechanism where contractors attempt to optimize a task allocation by successively selling and buying tasks in several trading rounds. Matrix auctions (MA) are truth-revealing and—in contrast to the Vickrey auction (VA) [24]—

applicable for the simultaneous assignment of multiple items or tasks to bidders. In a matrix-k-auction (MA-k), k items are auctioned-off simultaneously to some bidders. From their transmitted bids an auctioneer identifies the optimal allocation of all k items. Prices are set according to the VA, i.e. bidders receiving items pay the second-highest bid made for these items.

Empirical results on the suitability of these co-ordination mechanisms for the allocation of transportation tasks in a network of shipping companies are reported in [13]. In the experiments the mechanism ST, MA-2, and MA-3 showed promising performance for supply web co-ordination tasks.

4 Agent-Based Design of Holonic Manufacturing System

In this section we present the holonic design of a production node in a supply web. We assume that the production entity participates in the auctions as it was described in the overall framework of the last section.

There are already well-established layers of abstraction in the control of a flexible manufacturing system (FMS) (see Fig. 3): production planning and control (PPC), shop floor control (SFC), flexible cell control (FCC), autonomous system control, and machine control. Each of these layers has a clearly defined scope of competence. In Fig. 3 we can see holons on each of the five layers: at the lowest layer, the physical body of an autonomous system (i.e. an autonomous robot or a machine tool) together with its controlling agent. On the layer of the flexible cells we have the flexible cell control system together with the holons that are formed by the physical systems that belong to the flexible cell. On the SFC layer we have the agent that represents the SFC for a specific production unit together with all the holons that belong to it. Finally, on the enterprise layer we have all the holons that are present at a specific site of the company. Most of the holonic structures which were just described are quite stable. However, especially on the layer of the flexible cells it is very important to have efficient mechanisms to dynamically form new holons. When we try to describe this situation we have the conceptual problem that autonomous systems such as mobile robots might interact with flexible cells. To have in this case a more uniform view we assume that pure autonomous systems such as mobile robots and autonomous (guided) vehicles are represented as holons on the FCC layer, too. We refer to all these systems as flexible cell holons (FCH).

The main objective of the system is to efficiently process the tasks that are derived from the orders that come in when the production entity participates in the supply web. The enterprise layer derives form the orders individual production jobs that are announced to the SFC. The SFC produces a highlevel schedule for these production jobs that determines which job will be processed next on which machine tool. The SFC system again passes individual tasks (e.g. the transportation of a workpiece to a specific machine or the processing of a workpiece on a specific machine) to the lower FCC layer as soon as it is determined by the production plan that a task can be executed because all of the preceding steps in the working plan have been successfully completed. From these tasks

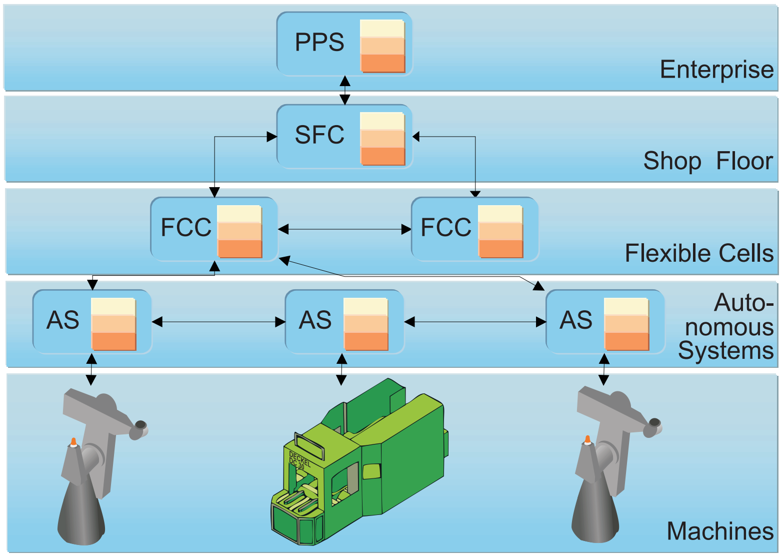


Fig. 3. Planning and control layers in a flexible manufacturing system.

the FCHs on the lower layers derive their local goals. The SFC system does not care whether it is possible for a group of FCHs to execute this task immediately or if they are currently engaged in the execution of a task. The SFC system just inserts the task into a list which is accessible to all of the FCHs and the FCHs decide by themselves when it will actually be executed. By doing a specific task, several FCHs have to co-operate. Each FCH has to play a part to solve a specific task. No FCH may believe that it is the only one that wants to play a certain part for a specific task. Therefore, the FCHs must co-ordinate their intentions to play parts in different tasks. The main problem to be solved is to find a consistent group of FCHs which together are able to solve a specific task. We call such a group a complete holon for a task. Only tasks for which a complete holon already is formed can actually be executed.

The FCHs can be separated into three groups: $\mathcal{R}$ mobile manipulation systems (mobile robots), $\mathcal{T}$ transport systems, and $\mathcal{C}$ flexible cells such as machining centres that might have locally fixed robots. In some settings even the workpieces which are to be processed are able to move autonomously, for example when they are installed on a transportation device. In these settings it is reasonable to control these workpieces by FCHs. We therefore introduce the set of workpieces $\mathcal{W}$.

Mobile manipulation systems are able to work flexibly on the given tasks. Each time a mobile manipulation system finishes the execution of a task it can start working on any task it is able to regardless of its current configuration. Locally fixed robots, machining centres, and flexible cells are much more restricted in their ability to choose tasks to be executed than FCHs in $\mathcal{R} \cup \mathcal{T}$ because FCHs

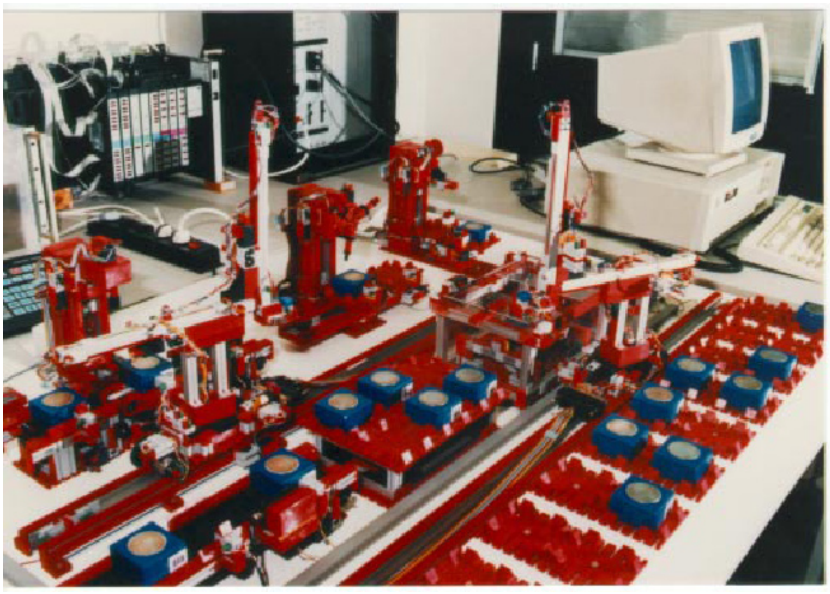


Fig. 4. A model of a flexible manufacturing system.

in $\mathcal{C}$ have a fixed location. An FCH f in $\mathcal{C}$ depends on FCHs of $\mathcal{R} \cup \mathcal{T}$ if all the devices needed for a specific task are not already present within f. We therefore introduce the precedence relations $\mathcal{W} \prec \mathcal{C} \prec \mathcal{T} \prec \mathcal{R}$ where $\mathcal{T} \prec \mathcal{R}$ means that a member of $\mathcal{R}$ may only join a holon for a specific task if all of the members of set $\mathcal{T}$ have already joined the holon for this specific task. The precedence relation $\prec$ is transitive which means that, for example, $\mathcal{W} \prec \mathcal{R}$ is valid too. The idea behind this definition is that the FCHs which are able to execute tasks flexibly may react to the decisions of FCHs which lack this flexibility in task execution.

To find a complete holon, the FCHs examine the list of tasks, which are announced by the SFC system, and try to reserve the task they would like to execute next for themselves. When an FCH is able to reserve a task successfully for itself, this FCH becomes the representative of the holon for this task. The representative r of a holon for a task t has responsibility to complete the holon for this task. r does this by sending messages to the other FCHs which ask these FCHs to join the holon for task t. A conflict occurs if two representatives send each other messages in which each of them asks the other one to join its own holon. It is possible to describe conflict resolution protocols for this situation which guarantee liveness and fairness of the system.

A prototypical holonic MAS has been implemented for the model of a flexible manufacturing system (see Fig. 4).

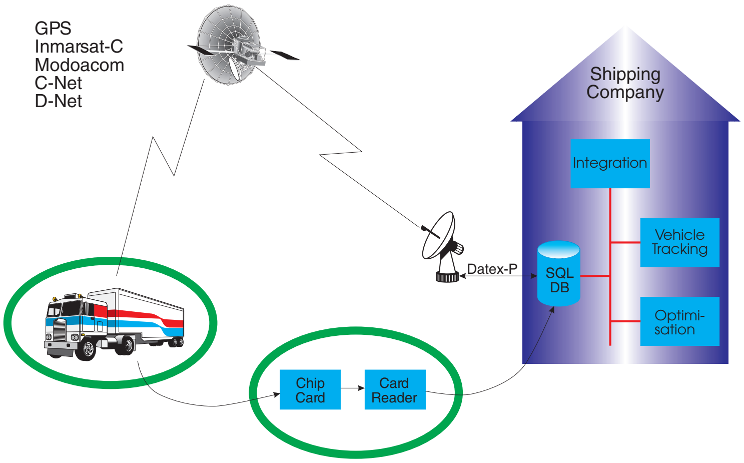


Fig. 5. The online fleet scheduling system TeleTruck.

5 Logistics Manangement in Virtual Enterprises: The TeleTruck System

This section describes the TeleTruck system which can be used for online order dispatching in a logistics management node (e.g., a haulage company) of a supply web. Again we assume that the logistics managment entity participates in the auctions going on in the supply web as it was described in the overall framework of Section 3.

A system that supports this setting has to cope with an online scheduling problem, in which at any point in time new orders can arrive and in which the system is able to react to problems in the execution of the computed plans. The TeleTruck system (see Fig. 5) implements an online dispatching systems using telecommunication technologies (e.g., satellite and mobile phone communication as well as GPS[3] information). [8, 9] demonstrated that a MAS approach to model an online dispatching system is feasible and and can compete with operation research approaches with respect to quality of the provided solution. However, in these scientific settings the transportation is done by self-contained entities. In practice we see that truck drivers, trucks, and (semi-)trailers are autonomous entities with their own objectives. Only an appropriate group of these entities can perform the transportation task together. For this reason a holonic approach had to be used to model the agent society of TeleTruck[4].

For each of the physical components (trucks, truck tractors, chassis, and (semi-)trailers) of the forwarding company as well as for each of its drivers there is an agent, which administrates its resources. These agents have their own plans, goals, and communication facilities in order to provide their resources for the

[3] Global Positioning System: Using such a system, a physical entity can find out its own position with an accuracy of a few meters.

transportation plans according to their role in the society. The agents have to form appropriate holons in order to execute the orders at hand.

Building a new holon is not just about collecting the needed resources. The components that merge to a holon have to complement each other and match the requirements of the transportation task. For each component an incompatibility list is represented that specifies the incompatibilities to other components, properties of components or orders. These constraints represent technical and legal restrictions and demands for holons.

For example an ordinary truck cannot haul a semi-trailer and a road train may not have more than five axes. Some of the constraints are hard while others are soft, i.e. relaxable. A truck driver can e.g. have several qualifications, such as specific licenses, language competencies for foreign orders etc. While a missing license for a truck is a hard constraint that forbids the combination of this truck and driver, language competence or the preference of a driver to drive "his" truck are soft constraints that can be relaxed or omitted.

The main things that need to be agreed between agents participating in a vehicle holon are to go to a specific place at a specific point in time and to load and unload goods. From these activities shared intentions for the agents participating in the vehicle holon can be derived. A new agent representing a *Plan'n'Execute Units* (PnEUs) for the vehicle holon is explicitly introduced to maintain the shared intentions of the vehicle holon. The PnEU coordinates the formation of the holon representing the transportation entity and plans the vehicle's routes, loading stops, and driving times. The PnEU represents the transportation holon to the outside and is authorised to reconfigure it. A PnEU is equipped with planning, coordination, and communication abilities, but does not have its own resources. Each transportation holon that has at least one task to do is headed by such a PnEU. Additionally, there is always exactly one idle PnEU with an empty plan that coordinates the formation of a new holon from idle components if needed.

For the assignment of the orders to the vehicle holons a bidding procedure is used [7]. The dispatch officer in the shipping company interacts with a dispatch agent. The dispatch agent announces the newly incoming orders, specified by the dispatch officer, to the PnEUs via an *extended contract net protocol* (ECNP) [8]. The PnEUs request resources from their components and decide whether the resources are sufficient to fulfill the task or not. If they are sufficient, the PnEU computes a plan, calculates its costs, and bids for the task. If the resources supplied by the components that are already member of the holon are not sufficient—which is trivially the case for the idle PnEU—the task together with the list of missing resources and a set of constraints that the order or the other members of the holon induce is announced to those agents which could supply such resources. These agents calculate which of their resources they can actually supply, and again announce the task and the still missing resources. This is iterated until all the needed resources are collected. The task of collecting the needed resources is not totally left to the PnEU because the components have local knowledge about how they can be combined, e.g., if a driver always

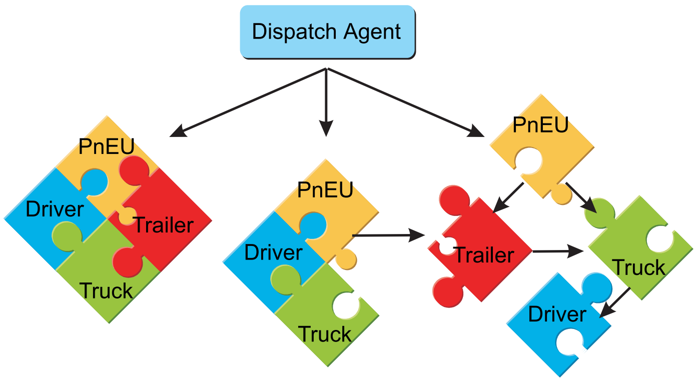


Fig. 6. Holonic planning in TELETRUCK.

drives the same truck it is local knowledge of the components and not of the PnEU.

Thus the ECNP is used on the one hand by the dispatch agent to allocate tasks to the existing vehicle holons and on the other hand by the free PnEU which uses the protocol to form a new holon. When the dispatch agent initiates the allocation of a transportation order to one or more vehicles, the semantics of the announcement is the invitation to the vehicles to bid for and execute a task. Fig. 6 shows a dispatch agent announcing a new transportation task to two vehicle holons and the idle PnEU. The complete vehicle holon on the left hand side cannot incorporate any further components. Hence, the head requests for the necessary resources from its components and, if these resources are sufficient, calculates the costs for the execution of the task. The second vehicle holon could integrate one further component. However, its head first tries to plan the task using only the resources of the components, the holon already has incorporated. If the resources are not sufficient, the head tries to collect the missing resources by performing an ECNP with idle components that supply such resources. The idle PnEU, which has not yet any resources on itsown, first of all performs an ENCP with those idle components that offer loading space; in the example a truck and a trailer. The trailer supplies loading space and chassis, therefore, it needs a motor supplying component. Hence, it announces the task to the truck. The truck which received two different announcements for the same task—one by the trailer and one by the PnEU directly—can bid in both protocols since it can be sure that only one of the protocols will be successful. Therefore, the truck agent looks for a driver, computes the costs for the two different announcements, and gives a bid both to the PnEU and to the trailer. Obviously, the costs for executing the task with a vehicle that consists of a driver and a truck are less than the costs of executing the same task with the same truck and driver and, in addition, a trailer. The idle PnEU will pass the bid of the truck to the dispatch agent. If the task is granted to the idle PnEU, the PnEU merges with the components to a vehicle holon and a new PnEU will be created for further bidding cycles.

Whenever the plan of a holon is finished the components separate and the PnEU terminates.

Since the tour plans that are computed in the ECNP procedure are suboptimal [8, 9], the *simulated trading* procedure [1] is used to improve the suboptimal initial solution stepwise towards globally optimal plans [9]. Simulated trading is a randomised algorithm that realises a market mechanism where the vehicles optimise their plans by successively selling and buying tasks. Trading is done in several rounds. Each round consists of a number of decision cycles. In each cycle the truck agents submit one offer to sell or buy a task. At the end of each round the dispatch agent tries to match the sell and buy offers of the trucks such that the costs of the global solution decrease. This implements a kind of hill-climbing algorithm. Like in the case of simulated annealing, a derivation that decreases from round to round can be specified such that in early rounds the dispatch agent is willing to accept a worsening of the global solution which is helpful to leave local maxima in the solution space. Nevertheless, local maxima are saved such that, when the algorithm terminates before a better solution is found, the best solution hitherto is returned. Hence, simulated trading is an interruptible anytime algorithm.

In order to allow the optimisation not only of the plans but also of the combination of components we extended the simulated trading procedure. It might be the case that a good route plan is not efficient because the allocation of resources to the plan is bad, e.g., a big truck is not full while a smaller truck could need some extra capacity to improve its own plan. We divided a trading round into three phases. The first phase consists of order trading cycles as explained above; in the middle phase the holons can submit offers to exchange components. The third phase is, like the first phase, an order trading phase. After the third phase is finished the dispatch agent matches the sell and buy and the component exchange offers. This final trading phase is needed to decide whether the exchange of components in the middle phase actually lead to an improvement of the global resource allocation.

6 Agent-Based Information Sources for a Software Repository

This section outlines how autonomous software agents can be used to collect information about software in a company that is geographically distributed to several sites. The main objective here is to keep consistency between the local installations at the different sites. The work presented in this section was done in collaboration with Dresdner Bank AG, a major German bank that has branch offices all over the world.

The availability of knowledge and information about internal business processes and software systems is strategically crucial and relevant for companies. One means to make such knowledge available in a smooth and easily usable way are repositories. They are computer-assisted information systems about the information processing activities of an enterprise. Other commonly used names are data dictionary, development database, information resource system, catalogue, or meta information system [21, 22].

The most important features of such a repository are the accuracy and recentness, as well as correctness of the contents made available. In order to keep the information about software systems in use up to date and in concurrence with the upgrading to new system releases and thus in concurrence with software configuration management procedures, we have implemented autonomous agents as pro-active information sources.

In the corporate setting of our project partner, the Dresdner Bank AG, software development follows strict rules. One of these includes the use of archiving servers for change management and data preservation. Here, software agents monitoring the activities on these servers provide a highly suitable means to communicate the most recent changes, releases, or patches to the corporate development repository in an autonomous and automatic manner. Software development inside the Dresdner Bank AG is performed in development centers, which are divided all over the world. In each development center, software development is done according to defined procedures and processes, where the software under construction, as well as the software which is in service, are collected by archiving servers.

The bank uses a broad amount of legacy software, developed for so-called host platforms (MVS-based mainframe systems). During the last few years however, with the introduction of client/server architectures and software development within this client/server environment, the need for change management has been tackled by a commercial tool, i.e. Continuus CM (CCM) [6]. In the client/server environment the CCM servers act as archiving servers which keep track of changes and releases. The development center is co-located within the intranet of the development repository. The software agents have been developed to find the most recent releases in the client/server environment by means of interacting with the CCM servers.

A general overview of the tasks and the flow of information for the agents in their working environment can be seen in Fig. 7. This figure also depicts the general requirements for the agents, as they were given initially. In the project development environment (right-hand side), a monitoring agent which interacts with the archiving server (CCM) is located, whereas in the so-called REPTIL[4] environment (left-hand side), we find a software agent for receiving new incoming information.

The communication and transfer of data in the network has to be done through secure channels, since corporate knowledge and values are transferred.

Archive-Based Agent in the Development Environment: We call this agent archive-based, since in the client/server environment the CCM server represents one specific kind of reference location to be monitored by software agents. In a

[4] The name Reptil has been chosen as acronym for the repository solution of the Dresdner Bank AG. The Reptil system will be commercially available in the near future.

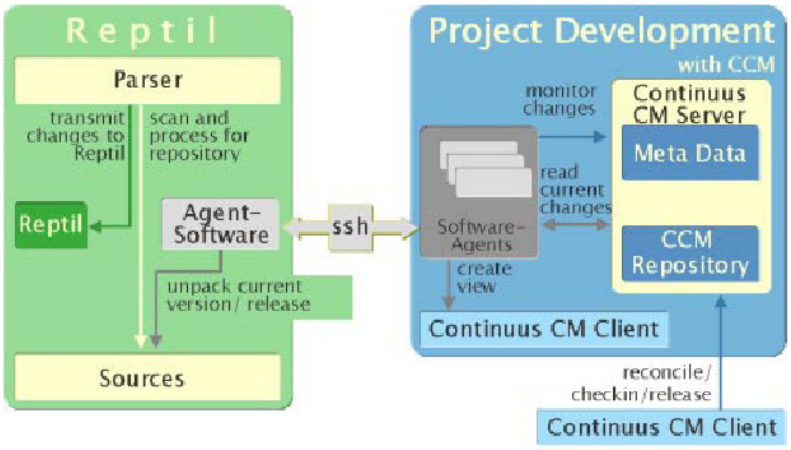


Fig. 7. Flow of information between development center (right) and repository Reptil (left).

more general view of the problem of transmitting the most recent release information as input data to the REPTIL system, such a reference location can be any kind of server or change management and tracking system, such as e.g. CVS [10], Rational ClearCase, CCM, or any other kind of interactive archive. The corporate choice of the bank was CCM, therefore our reference implementation currently interacts with it.

The agent software located in the development environment has to monitor the contents of the CCM repository in certain time intervals. It looks for specific tags on the CCM repository's contents. The CCM repository is organized in terms of (software) projects. Such a project contains all relevant information concerning the continuous evolution of a software system under development. CCM provides a broad amount of operations to keep track of changes and allows for concurrent development including features such as conflict detection. We will not go into detail about these features and operations. Instead we refer interested readers to the extensive system documentation [6]. In this article we limit our description of CCM features to the narrow requirements for the detection of new project releases, which is the task of the agent software.

A new release in the CCM project life cycle is characterized by a status label, which has the value released. Each object under CCM control has a type-specific life cycle, where the status label indicates its phase in the life cycle, for projects, the values of this label start with "working", and may include "test", "integrate" and finally "released" for making the software available. The agent's task is to find those projects, that have been released recently and are not yet updated or known in the REPTIL repository. When such a new release is detected, the agent creates its own CCM-specific project view and extracts the respective software and meta data available. The agent's general activity and interaction with CCM comprises the following steps:

- Monitoring of changes at the CCM servers;
- Upon detection of a new, unknown release, the creation of a CCM-specific project view, and
- the collection of the relevant sources code, as well as
- the compilation of available meta information, and finally
- packing and sending the data packages through the network to the REPTIL system.

The interaction with CCM has to be done in a way, that is transparent for the software developers using the respective CCM server, to ensure that the agent software does not change substantial system properties.

At the archiving server, we have designed and implemented an agent society, which consists of a collection of specialized agents. We model information extraction from the archiving server by task-specific agents. At the development site, there may exist more than one CCM server, which has to be monitored for relevant data. Therefore, we chose to separate the process of monitoring the archive, the data extraction from the respective archive as well as the data transmission.

The Repository REPTIL: The information system, that constitutes the in-house repository for the bank is called REPTIL3 (see Fig. 7). It processes, analyses and archives data about software projects and software systems in use. It makes this data available as information and meta-information to the users of the system. It also offers operations on the data in order to allow for knowledge transfer as well as information updates and refinements. Thus, the system is aiming towards providing a single point of information inside the software development processes of the bank.

The meta information which REPTIL offers to its users, comprises (among others) contextual knowledge about the development environment, such as software tools applied, as well as standards for procedures, user lists, lists of software components along with an analysis of their data and control flow, or organizational entities using the software and thus affected by changes to it. REPTIL provides users with a unified interface in a standard browser (e.g. Netscape or MS Internet Explorer) and is currently applied in the intranet of the bank.

While the agent software in the CCM environment actively seeks new project information and upon succeeding in this task, gets and sends it, the agent in the REPTIL environment has more simple tasks to achieve. It waits for new incoming data and upon receipt, unpacks the data and notifies the REPTIL system about it.

7 Conclusion and Future Work

In this article we have presented a framework for a supply web management system in which the individual nodes of the supply web are represented by autonomous software agents. The design of the coordination infrastructure for the supply web agents allows the individual nodes of the supply web to flexibly participate in auctions to acquire goods as well as to sell goods. This gives the participants in the supply web the ability to flexibly react to changes in the market situation. However, to actually benefit from this flexibility the internal design of the node in the supply web has to be able to internally cope with this situation. For the example of a flexible manufacturing system and a logistics management node the internal design was outlined in the article.

While most of the individual components of the overall framework are already implemented, the integration of this components is still work in progress. Future efforts will also be devoted to the integration of additional coordination mechanisms into the infrastructure. This aims to decentralize the configuration and coordination of distributed business processes as well as the (re-)allocation of resources and tasks within complex supply webs.

Acknowledgements

We would like to thank SAP Retail, espcially Hertel for their support of our research on agent-based supply chain management. We are also greateful to the Reptil project group at the Dredner Bank AG for their co-operative supoort in the work on agent-based software management as well as I. Zinnikus for design und implementation support.

References

[1] A. Bachem, W. Hochstättler, and M. Malich. Simulated Trading: A New Approach For Solving Vehicle Routing Problems. Technical Report 92.125, Mathematisches Institut der Universität zu Köln, Dezember 1992.

[2] A. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, Los Angeles, CA, 1988.

[3] Luc Bongaerts, László Monostori, Duncan Mc Farlane, and Botond Kádár. Hierarchy in distributed shop floor control. In *IMS-EUROPE 1998, the First Open Workshop of Esprit Working group on IMS*, Lausanne, 1998.

[4] H.-J. Bürckert, K. Fischer, and G. Vierke. Transportation scheduling with holonic mas – the teletruck approach. In *Proceedings of the Third International Conference on Practical Applications of Intelligent Agents and Multiagents (PAAM'98)*, 1998.

[5] J. Christensen. Holonic manufacturing systems — initial architecture and standard directions. In *Proc. of the 1st European Conference on Holonic Manufacturing Systems*, Hannover, December 1994.

[6] Introduction to continuus/cm. Continuus Software Corporation, 1998. see also http://www.continuus.com.

[7] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63 – 109, 1983.

[8] K. Fischer, J. P. Müller, and M. Pischel. A model for cooperative transportation scheduling. In *Proceedings of the 1st International Conference on Multiagent Systems (ICMAS'95)*, pages 109–116, San Francisco, June 1995.

[9] K. Fischer, J. P. Müller, and M. Pischel. Cooperative transportation scheduling: an application domain for DAI. *Journal of Applied Artificial Intelligence. Special issue on Intelligent Agents*, 10(1):1–33, 1996.

[10] K. Fogl. *Open Source Development with CVS.* CoriolisOpen(tm) Press, 1999.
[11] L. Gasser and M.N. Huhns. *Distributed Artificial Intelligence, Volume II.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, 1989.
[12] C. Gerber and C. G. Jung. Holonic Structures for Bounded Optimal Agent Societies. Technical Memo, DFKI, 1998. to appear.
[13] C. Gerber, J. Siekmann, and G. Vierke. Holonic multi-agent systems. Technical report, DFKI GmbH, 1999.
[14] P. Gomber, C. Schmidt, and C. Weinhardt. Efficiency incentives and computational tractability in the coordination of multi-agent systems. In *Proceedings of the Workshop Kooperationsnetze und Elektronische Koordination*, 1998.
[15] M.N. Huhns. *Distributed Artificial Intelligence.* Pitman/Morgan Kaufmann, San Mateo, CA, 1987.
[16] D. Kjenstad. Coordinated supply chain scheduling. Technical report, NTNU-rapport, 1998.
[17] A. Koestler. *The Ghost in the Machine.* Arkana Books, 1989.
[18] T.M. Laseter. *Balanced Sourcing: Cooperation and Competition in Supplier Relationships.* Jossey-Bass, October 1998. ISBN: 0787944432.
[19] M. Minsky. *The Society of Mind.* Simon and Schuster, New York, 1985.
[20] G. M. P. O'Hare and N. R. Jennings, editors. *Foundations of Distributed Artificial Intelligence.* Wiley & Sons, New York, 1996.
[21] E. Ortner. Repository systems. *Informatik Spektrum*, 22(4):235–251, 1999.
[22] E. Ortner. Repository systems. *Informatik Spektrum*, 22(5):351–363, 1999. In German.
[23] H. A. Simon. *The Sciences of the Artificial.* MIT Press, 6 edition, 1990.
[24] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
[25] Hans-Jürgen Warnecke. *Aufbruch zum fraktalen Unternehmen — Praxisbeispiele für neues Denken und Handeln.* Springer-Verlag, 1995.
[26] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT - Press, 1999.

Agent-Based Modelling of Ecosystems for Sustainable Resource Management[1]

Jim Doran

Department of Computer Science
University of Essex, Colchester, CO4 3SQ, UK
doraj@essex.ac.uk

Abstract. We present agent-based modelling and social simulation in particular application to ecosystem management. The steps in designing and building an agent-based model are discussed, as are the methodological problems typically encountered. Examples of relevant agent-based models are given. The task of integrated ecosystem management is considered and examples are given of agent-based modelling in this context. As a further illustration, consideration is given to a possible agent-based model of the Fraser River watershed in British Columbia, and to particular difficulties that it presents.

1 Agents, Multi-agent Systems, and Artificial Societies

Agents are entities that are autonomous loci of decision making. Agents sense, decide and act. *Reactive* agents choose actions by reference to their sensed circumstances, possibly with use of some kind of internal memory. *Deliberative* agents are those which, for example, reflect upon alternative courses of action, and select one of them for execution. That is, they plan. *Adaptive* agents change their behaviour in the light of changing circumstances. *Social* agents communicate and co-operate with other agents. All these various type of agents, and combinations of them, can be created in computer software to at least some degree.

There is a rapidly developing *agent technology* [24], a mixture of artificial intelligence and other computer science techniques, notably object-oriented programming. A range of alternative agent software design architectures have been developed, including architectures based upon AI planning systems, upon simple or "fuzzy" rules, upon artificial neural networks, and hybrid architectures.

A *multi-agent system* [44] is a set of interacting agents in some shared environment. Possible relationships between agents include co-operation and negotiation, task-delegation, subordination, and competition. Experimental inter-agent communication languages have been developed based on speech-act theory, notably KQML and FIPA-ACL. *Artificial Societies* are multi-agent systems that are engineered to display some of the features of human or other societies such as collective behaviour and co-operation.

The concept of *emergence* is prominent in multi-agent systems theory and practice. Emergent behaviour in a multi-agent system or artificial society is large-scale

[1] Earlier versions of some sections of this article appeared in [9] and [11].

behaviour which is significant, but which would not easily be predicted from a study of the individual agents within the system. Many natural phenomena appear to be emergent in this sense.

2 Agent-Based Modeling and Social Simulation

Agent-based modelling is essentially computer simulation, but using complex models that include agents [32]. Agent-based social simulation [16,17] is this use of agent-based model to simulate some aspect of animal or human society. In agent-based models of components of human societies, agents are often interpreted as in correspondence with individual humans, with small human groups, or with organisations. Thus agent-based models are not necessarily *individual-based.*

At its most successful, modelling (not necessarily agent-based modelling) promises major insight into the system modelled, the *target system,* leading to scientific advance, and useful prediction of the target system's behaviour leading to practical policy guidance. Even when, as is all too often the case, these primary objectives are only partially achieved there may be other "soft" benefits of a modelling study, for example the collection of data and the recognition of existing gaps in knowledge, enhanced communication between researchers in the domain, and education. Minimally, a modelling study will illustrate the concepts that inform and guide it. And quite often an attempt at modelling bring forward one or more interesting technical problems, which then embark upon a life of their own.

However, the further and particular promise of *agent-based* modelling in the social sciences is that it enables us to work with computer models that explicitly get to grips with the interaction between individual and social cognition. More specifically, that explicitly address the interaction between the real world and the collectively, that is socially, conceived world. Few would seriously deny that the beliefs and goals of individuals often influence and are influenced by the structure and dynamics of human society. Agent-based models with non-trivial agents within them enable such matters to be addressed.

3 Building and Using an Agent-Based Model

Let us first review standard computer simulation procedure, as it straightforwardly applies to agent-based modelling.

Formulation of Objectives/Questions. The remarks of Section 2 notwithstanding, the first step in the standard simulation procedure is to determine the goal of the modelling or simulation exercise as precisely as possible. There are various types of modelling goal. Is the aim to understand more about the target? to optimise the target? to verify a design for the target? But if possible the objectives should be stated much more narrowly than these broad categories. The more precisely they are stated, the more focussed the whole study can be, and the more economically the desired results can be obtained.

Design of the Model. Next it must be decided what structures and level of abstraction ("granularity") are to be used in the model. In particular, what agents, what particular types of cognition and what inter-agent communication are needed? What stochastic components are to be used? How is the shared environment of the agents to be expressed and what external influences are to be represented?

A key principle is to build the *minimum* into the model that is sufficient to answer the questions at issue.

Justification of Assumptions. Each of the particular assumptions built into the model, numerical or structural, needs to be empirically supported or, at the very least, inherently plausible. Collectively the assumptions must be coherent.

Choice of Measurements. Alongside the task of designing the structure of the model is the issue of what measurements should be taken as the model is being used. Ideally these will straightforwardly be determined from the modelling objective.

Choice of Software. Not all software systems are the same. Different systems and software platforms facilitate different model components and operations, although most enable a programmer to drop into a general implementation language as may be needed. Thus the choice of software should reflect the nature and objectives of the modelling study planned, not merely the particular computing facilities to hand. Well known software systems which may be used to support agent-based social simulation are CORMAS [7] which is specifically targeted at modelling for renewable resource management, SDML [40] which is rather more oriented to economics and organisation models, and SWARM [41] which has more of an "artificial life" pedigree. Other relevant software resources may be found listed at the CRESS[2] site [8].

Implementation of the Model within the selected software. Once the choice of a software platform has been made, the issue becomes how to implement the model within it. Initially the model will exist in its designer's head and as a natural language description possibly backed up by formal notations e.g. ERDs and State-Transition diagrams. Hopefully the chosen software will provide some form of high level model specification language or tool, which will facilitate the implementation task. As always in AI design implementation, there will be many further detailed design choices to be made.

Verification of the Model. Once implemented, the model must be verified, that is, the model must be checked to ensure that it is implemented correctly and as intended. This is largely a matter of checking the accuracy of the programming, and thus is in principle susceptible to standard techniques, but the likely use of pseudo-random number streams makes for added difficulty in detecting and tracking down errors.

Validation of the Model. The next step is to validate the model, that is, to check as far as may be possible that its behaviour is indeed in correspondence with that of the

[2] Centre for Research on Simulation in the Social Sciences

target. This requirement is distinct from, and in addition to, the earlier requirement that assumptions built into the model should be empirically supported or plausible. Validation involves systematically checking the model's actual observed behaviour against what is known of the target's behaviour.

Experimental Design and Sensitivity Analysis. When the model is established as valid, that is as reliable, then it can be put to serious use. It will presumably be the case that the original objectives of the study dictate certain investigations -- e.g. systematic "what if" questions. The theory of experimental design may be needed, especially if the model has substantial stochastic components. Sensitivity analysis is likely to be very important. For example, if the goal is optimality then there has to be sufficient exploration to identify and avoid false minima. Further we need to know what the optima critically depend upon.

Interpretation and Presentation of Results. The final stage in the standard procedure is to draw conclusions from the results and to decide how far they have general significance. There is, of course, always a great temptation to over-generalise.

4 Examples of Agent-Based Social Simulation

We now briefly describe three recently reported examples of agent-based social-simulation projects, that are representative of agent-based modelling in general and also relate to environmental management.

Carpenter, Brock, and Hanson [5] report a set of three agent-based models of a typical lake subject to phosphorous pollution and eutrophication. Each of the models also captures something of the physical, agricultural and social environment of the lake. The models are expressed in terms of difference equations that specify the lake, runoff etc., and seek to capture self-interested, rational decision making by farmers and others given their partial and possibly incorrect information. The three different models address different types of management regime: management by *market manager* (assuming a market in latest pollution data), management by *governing board* (an elected board, whose members differ in their views of "nature" and how to manage it), and management by *land manager* (who sets targets, regulations and incentives).

Because no specific lake is targeted, the conclusions reached are presented as indicative only. Significantly, the experimenters observe "irregular oscillations among ecosystem states and patterns of agent behaviour" for a wide range of scenarios. One particular conclusion they reach is that there is a "need for experimentation, learning and adaptation" by real managers (see Section 6, *adaptive management*).

Moss [31] reports an agent-based model of a specific organisation (North West Water, UK). The model, implemented in SDML [40], is directed to enhancing understanding of the company's management of critical incidents, and was developed after detailed interviews with company employees. The agents are based upon theories of cognition drawn from cognitive science and are, in a limited way, capable of selecting between simple cognitive rules (that are rather confusingly called "models" in the published account of the work) which they use to characterise and respond to a sequence of events. The agents are also adaptive in that they adjust the weight that they give to individual cognitive rules. A particular result of the study is to discover the importance of transferring information between successive shift managers.

The agent-based model of Kohler *et al* [26] is of a slightly different type. These authors have used their model to simulate the development of a Pueblo population settlement pattern from AD 900 to 1300 in a particular region of southwestern Colorado, USA. The agents in the model are rule-based and correspond to households. They take decisions about relocation and crop planting based upon such factors as local soil conditions, water resources, and recent weather. The simulated individual people in the households take no decisions as such, but are born, reproduce and die. From time to time these events lead a household to generate a new one to be located elsewhere. The model is implemented in SWARM [41]. Current work is enabling households to engage in product exchange as a medium term survival strategy. The output of the model, for alternative combinations of parameter settings, is being compared with a detailed archaeological reconstruction of the actual settlement pattern as it developed in prehistory. Unlike the two previously described studies this work is not directed, even in the longer term, to improvement of any management practice. Rather, it is targeted at enhanced scientific understanding

5 Methodological Problems of Agent-Based Modeling

The great increase in complexity in a model that follows from the use of deliberative or even reactive software agents within it brings substantial methodological problems in its train. We now discuss a number of major problems typically encountered in practical agent-based modelling work, with illustrative reference to the projects described in Section 4.

5.1 Skills and Time Requirement

The task of designing, constructing and experimenting with an agent-based model is a demanding one. Specialist expertise is required in the technical domain of the model (e.g. in artificial intelligence, agent technology, or difference equations) in order to make and carry through the detailed choices required. The time to program, debug and then experiment with the model is likely to be very substantial. The implication is that a multidisciplinary research team will be needed for such work, that brings together both the technical expertise required for the creation and handling of the actual model, and the social science expertise needed to analyse the target system. This need is amply illustrated in the projects described above, most obviously in that of Kohler *et al.*. Unfortunately, the required modelling expertise is sometimes downgraded in the belief that it is subordinate to the social science content, leading to shallow and ill-conceived models that merely act as a sounding board for a particular social science perspective.

5.2 Which Type of Model?

There are alternative types of model even within an agent-based formulation. The term "agent" itself is somewhat ambiguous. For example, the detailed technical content and connotations of the agents in the various models described earlier are quite different. Carpenter *et al* put the emphasis on rationality and cost-benefit analysis and design their agents accordingly. Moss adopts an agent design derived from a particular tradition of work in cognitive science. Kohler *et al* are content with a rule set for each agent representing a household. There is little guidance available in the literature on how to choose between such agent design alternatives in any reasoned way and investigators seem to rely largely on their disciplinary background. This is surely somewhat unsatisfactory, but it is hard to see how matters can be improved before the development of a reliable theory of agent design.

5.3 What Level of Abstraction?

Selecting agent type is an aspect of the more general problem of selecting for a model a level or combination of levels of abstraction, a problem that is especially difficult when emergent social phenomena are involved. It is never possible to construct a model that reflects the target system in comprehensive detail. Always the level of aggregation and abstraction must be chosen, with the aim of achieving tractability without significant loss of relevant model function. For example, even when a choice has been made of agent type it remains to be decided which elements of the target system are and are not to be individually expressed by agents. The answer may be many or few. Thus the Kohler *et al* study models individual households but not the cognition of individual people. The Carpenter *et al* models do not distinguish individual farmers and their families nor represent individual fish in the lake.

As with choice of model type, there seems to be rather little guidance available on how to choose the level of abstraction in a reasoned way. The standard procedure is to identify software agents with individual people or with clear-cut organisations or groups of people such as nuclear families. It is rare for any subtler notion of an agent correspondence to be considered.

A promising approach that has occasionally been explored, and which needs further work, is to define a lattice of models with levels in the lattice corresponding to levels of abstraction. Then criteria may be specified and algorithms defined which search over the lattice to locate an appropriate model for some particular modelling task. A basic requirement is that a model should be the simplest which will address usefully the questions to be asked of it. Lee and Fishwick [29], for example, have recently developed and discussed a dynamic model abstraction scheme and Fishwick, Sanderson and Wolff [15] have discussed its application to ecosystem management.

5.4 Searching a Massive Parameter Space

The complexity of the cognitive structure and processes of non-trivial agents implies a wide range of adjustable parameters and a correspondingly massive parameter space for an agent-based model. This is evident in all three of the projects described earlier. Furthermore, there will be many detailed structural choices. For example, agents that are to construct and execute plans of action, or which are to be adaptive, can be engineered to have these necessarily complex properties in many different ways and to many different degrees. How do we decide which model variants to investigate? How do we find where in the potentially enormous space of parameter settings and structural choices are the significant and interesting phenomena (if anywhere)? The likely presence of a degree of non-determinism in the model makes the task all the more daunting.

At present, this challenge is commonly avoided rather than met, with experimenters tending to assume that they know in advance which (few) adjustable parameters are and are not of importance. Arbitrary truncation of the space, and simple methods of systematic or random sampling, are clearly insufficient. There is a need for techniques and tools of experimentation incorporating heuristic but effective search processes, for example *hill climbing*, and methods of sub-region characterisation (e.g. [18]). Furthermore, we must keep in mind the possibility of discontinuities or singularities in the parameter space, and the possibility of chaotic model behaviour. There may well be points or regions in which a model's behaviour is effectively without regularity.

5.5 Implementation Issues.

A related question is the sensitivity of a model's behaviour to details of its computer implementation. A number of authors have suggested that the published results of modelling studies can sometimes reflect low level and insignificant implementation decisions rather than dynamic properties of the model of real scientific interest. An example is how to simulate concurrent activity on a single processor machine, where different implementation choices can have very different consequences (see e.g. [28]). There is little doubt that we still have much to learn here.

5.6 The Problem of Validation

Finally, and perhaps most important of all, there is the problem of model validation, that is, how to establish that a model does indeed reliably and usefully correspond to reality. As stated earlier, the traditional requirement has been that the model should be both inherently plausible and "checked out" against the target system in sufficient detail to establish confidence in the reliability of any findings made using it. This checking might either be at the level of the model's individual components, or at the level of its behaviour as a whole. But validating an agent-based model is particularly difficult because of its complexity of structure, especially cognitive structure. In fact, it becomes near impossible. For example, it is not difficult to dispute the detailed formulation of the difference equations that constitute the models of lake pollution described earlier, and the Kohler *et al* model of household decision making is clearly quite unreal as regards the actual process of decision making.

One reaction to this problem is simply to do the best one can given available knowledge (as Kohler and colleagues do), and another is to exhibit for some form of definite psychological validity for the agent structures (as in the Moss work). Or validation may be entirely in terms of a demonstrated match between a sample of the model's behaviour and the real behaviour of the target system. Whichever of these roads is followed, the study's conclusions are likely to be limited to broad generalisations [9].

Alternatively, we may discard the notion of modelling some specific target system (as do Carpenter and colleagues) and instead focus on the discovery of social theory [13]. In this case the emphasis is placed on the use of the model for reliable and objective derivation of the consequences of certain structural assumptions and certain parameter settings. Relevance is claimed to some class of social system rather than to any specific and actual system [9].

5.7 Summary Remarks on the Methodological Problems

The various methodological problems associated with agent-based modelling are substantial and of great practical significance. Their impact is apparent in current work. In the past, their cumulative effect has been to limit the effectiveness of agent-based modelling to little more than illustration of what *might* be possible and of how that potential *might* be realised.

6 Ecosystem Management

Environmental concerns have become ubiquitous as global human population increases along with its consumption of natural resources. Explicit and large-scale management of natural resources -- agriculture areas, forests, fishing stocks -- has been forced upon us as we attempt to solve ever-worsening problems of resource decline and failure. Notions of *integrated* management, covering all aspects of an ecosystem, and of *sustainability*, seeking to manage natural resources in ways that secure the future, have become prominent. The precise meaning of *sustainable* is somewhat elusive. A well-known, and cautiously worded, definition of sustainable development is "development that meets the needs of the present without compromising the ability of future generations to meet their own needs" [43, page 43]. Alternatively: "Sustainability means living within the constraints imposed by a finite set of global resources while satisfying the reasonable social and material aspirations of most of the world's citizens." [22]. It is commonly held that sustainability implies not only the preservation of the resource base, but also the achievement of fair access to it.

Over the past decade a widespread response to perceived past failures has been to reject centralised, "top-down" and "bureaucratic" management, in favour of *decentralisation* and a much more co-operative and community centred form of management. Since management and its failures impacts peoples' lives in profound ways, social and political reverberations are inevitable. Indeed, certain styles of government and multinational commerce and industry have become major targets of criticism, hostility and even violence.

A number of ecosystem management strategies are prominent in the environmental resource management literature, and they reflect the current emphasis on decentralised and co-operative ecosystem management. They include:

- *Maximal Stakeholder Involvement*
- *Co-management*
- *Community Management*
- *Adaptive Management*

These different strategies may be summarised as follows:

> *Maximal Stakeholder Involvement* [12,19] addresses a situation in which there are multiple competing stakeholders for a particular ecosystem e.g. various branches of government, non-governmental organisations, and corporations. It is argued that stakeholders must be drawn into discussion, negotiation and co-operation. This is partly to achieve better decision making by way of better mutual understanding and shared knowledge, but also to achieve greater commitment by the stakeholders to the actual implementation of solutions.
>
> *Co-Management* [25,33] advocates collaboration between government agencies and user groups, especially in the context of fisheries management (a "common property" resource) and especially for regulatory decision making. The aim is to do away with what is seen as "the distant, impersonal, insensitive, bureaucratic approach" [25, page 424] of central government responsibility and decision-making, and to replace it by an effective and dynamic partnership.
>
> *Community Management* [35, 6, 27] goes a step further than co-management. It seeks to draw upon (or return to) local community structure and local "social capital" [36] as a basis for resource system management. Its emphasis is upon the effectiveness of local knowledge and of emergent community self-regulation in contrast to attempted direction by central government. This strategy is advocated particularly in the context of non-industrialised countries.

By contrast:

> *Adaptive Management* [42,21] is more concerned with the "how" of management than the "who". The emphasis is on a (new) management style in which experimentation is at the heart of management practice so that information is gathered about the ecosystem as management proceeds, and action is varied accordingly.

Notice that these existing strategies typically put the emphasis on the management regime to be achieved, rather than on the intervention process by which it is to be achieved. The intervention process itself is assumed to be a matter of contact, discussion and persuasion on the part of the specialist team concerned, which might be anything from a lone doctoral student to an NGO, governmental or even United Nations team.

A further and rather different possibility is merely to act to *improve the resource sub-system* (for example by cutting a new canal, or introducing a new and better yielding crop variety), without seeking to change, or even understand, the associated human socio-cultural system. However, there is much evidence to suggest that this approach can easily prove counter-productive [36].

These various management strategies, although all in practical use, are not all of the same type, are not precisely defined, and are not exhaustive or well differentiated.

Their limits of applicability are little considered in the literature except in the vaguest terms. The question naturally arises, therefore, which of these strategies should be used in which circumstances? And what other possible ecosystem management and intervention strategies are there?

6.1 Centralisation and Decentralisation

Independently of the actual choice of intervention strategy, the execution of any particular strategy may often be divided into two steps. These steps are first the collection of information about the agent hierarchies and the design, if appropriate, of a revised organisation for them, and then secondly the actual process of intervention to bring about the chosen organisation and to set it in operation. Choosing an organisational structure for agents prompts consideration of centralisation and decentralisation and criteria for their effectiveness. Accordingly we now briefly discuss the impact of hierarchies on collective task performance.

From a computational perspective, centralised hierarchical organisation potentially enables one agent (or a small number of agents) to take and execute decisions in the light of a single abstract overview of the task scenario, deploying appropriate abstract knowledge. This may be of crucial benefit to efficient collective task performance. But this benefit will not arise if

(i) the task simply does not require such an overview for its performance (e.g. if local problems do not inter-relate), or if
(ii) no adequate abstraction scheme exists, or if
(iii) the flow of information up the hierarchy and of action down the hierarchy frequently fails or is too slow.

Point (ii) requires some elaboration. If there is no effective method available to the hierarchy of abstracting and summarising the task and the knowledge required to manage it, then any attempt at centralised overview and management is doomed to failure or, at best, overload in cognitive processing.

These insights, although they originate in the study of artificial agent societies, must surely also apply to human societies. They suggest that the objective of *decentralisation* should be to recognise and address these difficulties with centralisation, insofar as they exist in the particular case, by striking a balance between the actual need for and feasibility of central overviews and the delays and errors inherent in any substantial attempt to create them. However, it sometimes seems that the present move towards decentralised management is *not* grounded in the need for this balance, but is rather at root no more than an emotional and ideological reaction in the face of a global problem that human society is not currently equipped to address.

7 Modeling for Ecosystem Management

In the past formal ecosystem models have most commonly been formulated mathematically, for example, as a set of simultaneous linear equations (e.g. the model of the Prince William Sound ecosystem after the 1989 Exxon Valdez oil spill [34]) or more often as differential or difference equations [45]. A recent and striking example of the latter is the work of Anderies [2] who has developed new models of the Tsembagan (New Guinea) and Easter Island ecosystems, including their human components. These models are economics based, and be fit well the available evidence. The Easter Island ecosystem is particularly interesting. It displays, in the years since 400 AD, just the uncontrolled exploitation of a natural resource by a human population, leading to disaster, that is now in prospect at the global scale and that concerns so many. Anderies' model is an effective tool with which to investigate this dramatic ecosystem history.

7.1 Why Use Agent-Based Models?

As we have stressed, using agent-based models adds new power to the modelling process. Thus Anderies himself recognises that his non-agent-based approach "ignores important heterogeneity across agents that would be captured in a multi-agent model" [2, page 401]. And furthermore he draws attention to the need to address the emergence of institutions in a context where it is institutions that may or may not succeed in regulating resource management [2, page 410]. These observations highlight two important advantages of agent-based models: the ability to handle heterogeneous agents, and the ability, above all, to handle the cognition which underlies so much of social activity and structure, including the dynamics of institutions and social organisation.

7.2 Examples of Agent-Based Models

Agent-based models involving environmental factors have been experimented with in some social disciplines for many years, although often without use of "agent" terminology (for their particular use in archaeology, see [10]). Three examples of agent-based simulation studies with an environmental dimension were described in Section 4. Other substantial programmes of work have targeted (a) modelling of natural resource management in under-developed communities with a particular emphasis on stakeholder involvement on role-playing scenarios [3,4,38] and (b) multi-level modelling of complex ecosystems with a strong object-oriented flavour, [15,29]. The ongoing European Community funded FIRMA project [14] is dedicated to an agent-based approach to water management.

7.3 Types of Agent-Based Models

We may usefully recognise different types of agent-based models of an ecosystem, of increasing degrees of complexity. The computationally simplest possibility is merely to *conceive* of the model in agent terms, without reflecting that conception in the software at all. Beyond this computationally trivial case (which is, however, not infrequently met) are agent-based models of only the non-human components of an ecosystem. It follows that the agent architectures involved will be relatively simple. A further step adds into the software explicit structural representation of human beings and the human social component, beginning therefore to address the complexity of human thought and the interaction between human society and its environment in the ecosystem. The most complex model type goes further and focuses on the management regime that is or needs to be applied to the ecosystem, and on how this management regime may be modified or brought about by a controlled process of *intervention*. To illustrate this last and most complex type of model, and the problems associated with it, we now consider integrated watershed management.

8 Managing Watersheds

Integrated river watershed management is a category of ecosystem management that is of obvious importance and complexity (see, for example, the various studies of the Nile watershed presented in [1], and the review of watershed modelling by Westervelt [45]).

Potentially conflicting requirements within a watershed are:

water supply (for domestic, agricultural and industrial uses)
pollution control
fisheries management
flood control
hydropower production
navigation and wetlands management
recreation provision

Biophysical investigation addresses such matters as water flows, fish population dynamics, types and levels of pollution, and the impact of control procedures. But beyond problems at the biophysical level are important human social problems that are sometimes insufficiently considered.

Human society in a watershed typically comprises a multiplicity of organisational *stakeholders*, each pursuing its own interests, and each with only limited awareness and detailed knowledge of the others. There may well be sharp political and cultural differences. Put in general terms, the task at all scales in a watershed is to execute a strategic program of natural resource utilisation and conservation which balances sectional interests and which is sustainable. To achieve sustainable balance is not easy, and is impossible without restraint and compromise. Again expressed in general terms, a number of particular sources of difficulty are apparent. These include the complex structure of the environmental subsystems, the private agendas of the particular stakeholders and their conflicting norms and expectations, stakeholders' lack of information, and failures of inter-stakeholder communication.

It therefore seems that if agent-based modelling is to be used to address watershed integrated management problems, models must address the social and human issues of data integration, partial knowledge, decision-making, conflict resolution, and negotiation. In sum, we must focus on *social intervention strategies*. Agent-based modelling is of potential value just because it can plausibly extend computer modelling into this arena.

9 The Case of the Fraser River Watershed

To get a feel for what is involved in integrated watershed management we look in a little detail at an important particular case, the Fraser River in British Columbia, Canada.

The management task for the Fraser may be viewed on any of three scales: the entire Fraser watershed; the Lower Fraser Basin and separately, the Upper Fraser; or, recursively, any of the sub-watersheds of the Fraser. At each scale there are particular characteristics and problems as well as the problems common to all three scales.

The Fraser River watershed as a whole is large and diverse ranging from conurbation (Greater Vancouver) to near wilderness. Logging is a major industry, and the salmon populations are of major importance but in serious decline. The watershed displays all the aspects and problems mentioned earlier. Over the past decade a major attempt at integrated management of the watershed at this scale has been made, but this has encountered many social and political difficulties and as yet has had limited success [12,30]. A particular problem is the complexity of governance. Federal, Provincial, Municipal and First Nations governments are all stakeholders in the watershed. The aboriginal or "First Nation" dimension adds further distinctive complexity as aboriginal culture becomes more influential and, for example, land rights are (re-)negotiated.

The Lower Fraser Basin [23] has experienced a population explosion over the past century, which has resulted in de-forestation, agriculture, industrialisation and conurbation, and rising levels of pollution. There is now a major immigrant population, a majority of Chinese race, whose cultural norms and expectations do not fully align either with those of European or of First Nation origin. This ethnic diversity further complicates the environmental issues described earlier.

The particular problem of the smaller urban sub-watersheds, for example, the Brunette River watershed [19] is that any attempt at integrated management must grapple with a range of powerful stakeholders, including any or all of the four orders of government listed earlier, and major corporations. Achieving a coherent watershed management plan (and financing and then implementing it) becomes a delicate matter of initiating and bringing to fruition negotiation between the representatives of these powerful stakeholders, with local community input where possible.

In sum, study of the Fraser watershed reveals a diversity of stakeholders, the majority organised into decision-making hierarchies, that are currently *failing* to collaborate to solve both watershed wide and more localised distributed environmental control problems, with the result that key environmental variables are moving out of their acceptable ranges.

10 An Outline Agent-Based Model for the Fraser Watershed

We now set out to specify in outline a model of a watershed, such as that of the Fraser, and of ways in which intervention might establish an effective management regime for it. The model to be described must, of course, ultimately be programmable for a computer. This is a fundamental requirement. The model is also highly abstract.

The aim is achieve new and more precise understanding of the typical intervention scenario, rather than to draw specific conclusions about the Fraser watershed in particular. Finally, the model is individual-based, with a very loose correspondence between people and software agents.

The model takes as its start point the assumption that there exist centralised hierarchies of agents, with overlapping spheres of action, which are competing in an initially resource rich environment. In a little more detail:

1. *There is a large society of software agents (>1,000). The society is dynamic in the sense that there are ongoing actions by agents and interactions between agents, and possibly changes in social organisation of the society from time to time. The agents are heterogeneous in the sense that different agents have different behaviours and beliefs, and different types and degrees of cognitive ability.*

2. *The agents are organised into hierarchies which are, in effect, competing. At the top of each hierarchy is an agent (or are agents) whose goals reflect the resource interests of the hierarchy as a whole. Agents lower in the hierarchy normally act loosely in accordance with more limited goals set by their immediate superiors.*

3. *The society is set in a simulated natural environment incorporating an abstract resource management task that impacts all agents. Solutions may be assumed typically to require extensive agent co-operation and compromise. The hierarchies have overlapping spheres of activity.*

4. *Intervention is possible in the sense that the "experimenter" may at any times send messages of specified types to some or all of the agents within the society.*

To take this model specification close to an actual computer implementation, we now comment on each of the four components in turn.

The agents. As has been emphasised, agents are software entities. There exists a range of available agent designs that might be adopted for a model such as this, but none entirely suitable. The simplest agent architectures use a small number of *condition-action rules* to determine action. However, in this context it is hard to avoid the conclusion that agents should include beliefs, goals and, to some degree, internal models of their "social" context and the environment. They need the ability to act, perhaps via the creation and execution of plans, and the ability to send and receive messages. One way to express these requirements is to say that agents must have an operational *Belief, Desires and Intentions (BDI)* architecture [44, page 585], including the ability to process incoming and outgoing speech-acts. Such agent architectures have indeed been developed, but only in experimental research contexts and with cognitive abilities still very limited compared with human beings. These architectures have rarely if ever been used for agent-based social simulation.

The agent hierarchies. There are many ways in which a set of software agents may be engineered to function as an organised, centralised hierarchy. For a variety of possibilities see [37]. Perhaps the simplest method is to include within each agent specific condition-action rules that straightforwardly lead it to play its required role within the hierarchy. This role will include responding appropriately to messages from subordinates, and the completion of tasks delegated to it from superiors (which may involve sub-delegation). The rules may also support messages being exchanged with other agents at the same level in the hierarchy. A major limitation of this rule-based agent architecture is that the agents are not themselves able to vary or withdraw from their own role, so that the hierarchy is inflexible.

A different and more "realistic" approach is to design agents in terms of the (self-interested) goals that they contain and seek to achieve. An agent's role within the hierarchy then flows from its goals and the plans that it creates and executes to achieve them. The agent decides to undertake the appropriate role tasks for the rewards that it believes will accrue to it by doing so. Clearly this requires that the hierarchy functions to deliver rewards to agents, or, at least, that agents believe that it does. This approach is more powerful, and the hierarchies formed more flexible, but is also much more complex because it requires that agents be designed and implemented to adopt, choose between, delete, and work to goals as a primary activity (c.f. the BDI architecture mentioned earlier). We assume that the agents in such a hierarchy will be heterogeneous in various ways, notably in the autonomous goals that they possess, the beliefs about the environment and the hierarchy itself that they possess, and their actual ability to plan and execute plans including plans involving others.

Another key aspect of any multi-agent system is inter-agent communication. In this context *ad hoc* communication is likely to be sufficient, but a standardised inter-agent communication language such as KQML or FIPA-ACL is also a possibility.

The environment sub-model. The natural environment sub-model must capture the need for long-term co-operation to achieve sustainability, with failure to co-operate by a significant proportion of agents leading to collective disadvantage [20]. Co-operation implies short-term restraint by agents, rather than "greed", and both local and global action by agents. The sub-model should also permit the issue of equality or inequality of agent access to resources to be addressed.

In a typical watershed there are many variables and stakeholders of relevance. Potentially significant variables include: input flows to tributaries, irrigation requirement and take, fish stock levels and catch, levels of use of agricultural chemicals, pollution levels, dam flow settings, hydroelectric power generation levels, dam & lake recreation requirements. Stakeholders include local and national government, farmers, fisheries and river conservationists, hydropower companies, and recreation organisations.

A suitable form for the environment sub-model might thus be set of variables and parameters divided into (overlapping) subsets as follows:

- independent *environmental* variables (typically stochastic e.g. an input flow)
- *sensed* variables (variables one of more agents can read, e.g. pollution level at a particular point)
- *resource consumption* variables (variables one or more agents can diminish e.g. consumption of water)

- *input* variables (variables one or more agents can augment e.g. amount of pesticide)
- *action parameters* (values one or more agents can set e.g. specifying the flow through a dam)

together with a set of non-linear recurrence relations, which specifies the interconnections between the variables. Spatial distribution would be implicit in the model structure. The *sphere of activity* of a particular agent (or agent hierarchy) would then be that subset of all the variables and action parameters that is accessible to it. Limited spheres of activity imply a need for inter-agent co-operation.

Put in the most general terms, the intervention task is then to influence the agents so that certain selected variables are kept indefinitely within specified limits, in the face of fluctuation in independent environmental variables. The agents can refer to sensed variables and can adjust those variables that they collectively have within their control. Just which variables are to be maintained within which limits (so achieving sustainability and equity between agents) is taken as inherent to the intervention task itself. Specific ecosystems, for example the Fraser watershed, would give rise to specific environment sub-models, with a consequent requirement for model validation. Fortunately, the search for effective interventions can, and arguably should, be made in terms of *typical* environment models.

The possible interventions. We are restricting ourselves to interventions that *involve no significant new resources*. This means that interventions take the form only of messages sent by the experimenter to agents in the model. Interventions that directly impact the natural environment sub-model are excluded. The *intervention language* should precisely specify the syntax and semantics of the messages the experimenter may send to agents. As in the case of inter-agent communication, the language could be *ad hoc* or based upon an established language. Messages might be limited to assertions and questions, or include such content as suggested goals and plans. Whatever intervention language is used, agents must be able to receive, assimilate and act upon -- or decline to act upon -- the allowable messages and their content. This may well involve complex issues of belief and goal adjustment, depending upon the complexity of the agents themselves.

10.1 Incompleteness of the Specification

This outline model specification is certainly not fully precise, and is in many ways arbitrary. To develop the specification to one of full precision would be a technically challenging task, and it would always be difficult to justify one model variant rather than another. For example, what type of internal representation (if any) of the agent hierarchy of which it is a member should an agent maintain and use? What range of inter-agent communications should be implemented? Furthermore, limitations of current agent technology mean that the complexity and cognitive powers of the software agents will be very limited, with all that that potentially implies for realism.

11 Using the Outline Model

11.1 Discovering Intervention Strategies

Assuming that the model (comprising agents and agent hierarchies, environment sub-model, and intervention set) has been formulated in detail and implemented on a suitable computer, the next task is experimentally to study possible interventions in search of those leading to sustainability and social justice. Essentially, an intervention is a single message sent to a single agent or "broadcast" to a particular set of agents. Within a precise model, the set of all possible combinations of interventions is well defined, assuming that the timing of the messages is specified in some suitable way. The performance of a set of interventions within *the model* may be assessed by their effectiveness as measured in the natural environment sub-model.

The discovery task involves repeatedly setting the model running, and then observing the effect of particular combinations of messages sent to the agents. Often there will be a degree of non-determinism, so that the same combination of intervention messages may yield different outcomes on different occasions even for an identical initial set-up of the model. Note that the "null" case, where the model is allowed to run with *no* messages sent, may itself produce highly complex behaviour and requires special investigation.

The set of all possible interventions is clearly extremely large. Indeed it is easily made infinite. The task of searching for effective combinations therefore seems close to intractable. However, much can be done automatically, at least in principle. Many relevant search techniques are available in the artificial intelligence repertoire, including varieties of hill-climbing search and of genetic algorithms [39]. Furthermore it is appropriate to try to work in terms of *intervention strategies*, that is, sets of interventions organised and targeted in some particular way, and then to reformulate the task as that of searching for effective intervention strategies. Working in terms of intervention strategies is attractive, but begs the question of their precise definition. Even if intervention strategies can be precisely formulated in terms of a specified intervention language (and this will not be easy), their effectiveness might depend greatly upon details of the agents, of the agent hierarchies, and of the resource problem. This would imply that there is no simple and general "map" of the exploration space to discover. At worst the impact of particular interventions might prove to be entirely unpredictable in all circumstances, but this would be counter-intuitive.

11.2 Testing Discovered Strategies

Suppose that we find an effective intervention strategy as judged by its performance within the model, that is, by its impact upon the agent society and the society's interaction with the simulated natural environment. How can the discovered strategy be put to the test and validated *outside* the model? There seem to be two main possibilities:

1. *Apply the discovered intervention strategy in real situations and observe the consequences*

2. *Systematically compare the effective strategy discovered in the model with standard intervention practice and seek insights, agreements and disagreements*

The first of these possibilities is clearly a major undertaking with substantial financial, social and political implications and risks. The second is much more immediately possible, but a degree of objectivity is lost since the assessment criteria become more subject to influence by current theoretical preferences and assumptions.

A third and interesting possibility is to use the agent model as the basis of an artificial role-play scenario in which stakeholders "play themselves" [3]. It seems questionable, however, whether stakeholders playing these roles could respond to interventions quite as they would in real life.

11.3 Discussion

The striking feature of this attempt to specify in outline a model for the Fraser watershed is the way in which the documented difficulty of achieving integrated watershed management for the Fraser unsurprisingly translates into a technically challenging modelling task. Actually implementing the model outlined here on a computer, in any non-trivial way, would require very substantial complexity and sophistication in the technical design, and it is far from clear that this is achievable in the present state of art. Thus it seems that this type of agent-based modelling *although just what is required in many contexts* should be seen as a spur to research rather than currently a practically usable tool.

12. Prospect

Agent-based social simulation is potentially a very powerful means to explore the dynamics and potentialities of ecosystems, including their human components. But there are many practical difficulties, as touched upon here and discussed in, for example [11]. These practical difficulties are substantial and cannot be ignored. There is no "free lunch". The consequence is that the "soft" uses of agent-based modelling described in Section 2 are likely to be prominent for some time to come. Research in agent-based modelling must surely be directed to how the methodological difficulties may be overcome or minimised, and go hand in hand with the more straightforward application of agent-based modelling to tackle particular problems using the available, if limited, technology.

References

1. Abu-Zeid M.A., and Biswas A.K. (1996) *River Basin Planning and Management*. Oxford University Press.
2. Anderies J.M. (2000) On Modeling Human Behavior and Institutions in Simple Ecological Economic Systems. *Ecological Economics, 35,* 393-412.

3. Bousquet F., Barreteau O., Le Page C., Mullon C. and Weber J. (1999) An Environmental Modelling Approach: the Use of Multi-Agent Simulations. In: *Advances in Environmental and Ecological Modelling* (Eds. F. Blasco and A. Weill). Paris, Elsevier. Pp. 113-122.
4. Bousquet F., Cambier, C., Mullon C., Morand P., Quensiere J. (1994). Simulating Fishermen's Society. In: N. Gilbert & J. Doran (eds.) *Simulating Societies*. UCL Press. Pp. 143-164.
5. Carpenter S., Brock W., and Ghanson P. (1999) Ecological and Social Dynamics in Simple Models of Ecosystem management. *Conservation Ecology 3(2): 4. [ONLINE]* URL http://www.consecol.org/vol3/iss2/art4
6. Christie P. and White A.T. (1997) Trends in Development of Coastal Area Management in Tropical Countries: From Central to Community Orientation. *Coastal Management, 25:155-181.*
7. CORMAS software system. URL http://www.cirad.fr/presentation/programmes/espace/cormas/eng/index.shtml
8. CRESS *Centre for Research on Simulation in the Social Sciences*, University of Surrey, Guildford, UK. URL http://www.soc.surrey.ac.uk/research/simsoc/cress.html
9. Doran J. E. (1997). From Computer Simulation to Artificial Societies. *Transactions of the Society for Computer Simulation International, 14,* 69-78.
10. Doran J. E. (1999). Prospects for Agent-Based Modelling in Archaeology. *Archeologia e Calcolatori, 10*, 33-44.
11. Doran J. E. (2000) Hard Problems in the Use of Agent-Based Modelling. *Social Science Methodology in the New Millennium. Proceedings of the Fifth International Conference on Logic and Methodology, October 3rd-6th, Cologne, Germany.* Published as CD-ROM.
12. Dorcey A. H. J. (1997). Collaborating Towards Sustainability Together: The Fraser Basin Management Board and Program. In: D. Shrubsole & B. Mitchell (eds.). *Practising Sustainable Water Management: Canadian and International Experiences*. Canadian Water Resources Association. URL http://www.interchg.ubc.ca/dorcey/chcwra/fccwra.html
13. Epstein J. M. and Axtell R (1996) *Growing Artificial Societies: Social Science from the Bottom Up*. Washington, D.C.: The Brookings Institution Press and Cambridge, MA: MIT Press.
14. The FIRMA project. URL http://www.cpm.mmu.ac.uk/firma/index.html
15. Fishwick P. A., Sanderson J.G. and Wolff W.F. (1998) A Multimodeling Basis for Across-Trophic-Level Ecosystem Modeling: The Florida Everglades Example. *SCS Transactions on Simulation*, 15(2), 76-98.
16. Gilbert, G.N. (2000) Modelling Sociality: The View from Europe. In: T. A. Kohler & G. J. Gummerman (eds.). *Dynamics in Human and Primate Societies.* Oxford University Press. Pp. 355-372.
17. Gilbert G N and Troitzsch K G, (1999). *Simulation for the Social Scientist*. Open University Press.
18. Hales, D. (2001) Memetic Evolution and Sub-Optimisation. PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK.
19. Hall K. *et al* (2000) Brunette Basin Watershed Plan. Policy and Planning Department, Greater Vancouver Regional District.
20. Hardin G. (1968) The Tragedy of the Commons. *Science, 162,* 1243-1248.
21. Healey M. (1998). Paradigms, Policies, and Prognostications about the Management of Watershed Ecosystems In: R. J. Naiman & R. E. Bilby (eds.) *River Ecology and Management*. Springer. Pp. 662-682.
22. Healey M. (1998a) Barriers and Bridges to Sustainability in the Fraser Basin. Invited address to the *State of the Basin conference*, November 20 & 21st, 1998. Vancouver.

23. Healey M. (ed.) (1999*) Seeking Sustainability in the Lower Fraser Basin: Issues and Choices.* Institute for Resources and the Environment, Westwater Research Centre, University of British Columbia, Vancouver.
24. Jennings N.R., and Wooldridge, M.J. (eds.) (1998). *Agent Technology*. Springer: Berlin.
25 Jentoft, S., McCay B., and Wilson D.C. (1998) Social Theory and Fisheries Co-Management, *Marine Policy.* Vol. 22, No. 4-5, 423-436.
26. Kohler T.A., Kresl J., Van West C., Carr E., Wilshusen R. H. (2000). Be There Then: a Modelling Approach to Settlement Determinants and Spatial Efficiency Among Late Ancestral Pueblo Populations of the Mesa Verde Region, U.S. In: T. A. Kohler & G. J. Gummerman (eds.). *Dynamics in Human and Primate Societies.* Oxford University Press. Pp. 145-178.
27. Lansing J. S. (2000) Anti-Chaos, Common Property, and the Emergence of Cooperation. In: T. A. Kohler & G. J. Gummerman (eds.). *Dynamics in Human and Primate Societies.* Oxford University Press. Pp. 207-223.
28. Lawson B. G. and Park S. (2000). Asynchronous Time Evolution in an Artificial Society Model. *[ONLINE] JASSS, Vol. 3, No. 1.*
29. Lee K. and Fishwick P. A. (1997) A Methodology for Dynamic Model Abstraction *Transactions of the Society for Computer Simulation International,* 13(4):217-229.
30. Marshall D. (1998) Watershed management in British Columbia: The Fraser Basin Experience. *Environments*, Vol. 25, No. 2/3, 64-79.
31. Moss S. (1998) Critical Incident Management: An Empirically Derived Computational Model. *[ONLINE] JASSS, Vol. 1, No. 4.* URL http://www.soc.surrey.ac.uk/JASSS/1/4/1.html
32. Moss S. & Davidsson P. (forthcoming). *Proceedings of the Second Workshop on Multi-Agent Based Simulation (MABS 2000), Boston, 8-9th July, 2000.* To appear in Springer LNAI series.
33. Nielsen J. R. and Vedsmand T. (1999) User Participation and Institutional Change in Fisheries Management: a Viable Alternative to the Failures of 'Top-Down' Driven Control? *Ocean & Coastal Management, 42:19-37*.
34. Okey T.A. and Pauly D. eds. (1998). *Trophic Mass-Balance Model of Alaska's Prince William Sound Ecosystem, for the Post-Spill Period 1994-1996*. The Fisheries Centre, University of British Columbia, Canada. ISSN 1198-6727.
35. Ostrom E. (1990). *Governing the Commons: The Evolution of Institutions for Collective Action.* Cambridge University Press: Cambridge, UK.
36. Ostrom E. (1995). Constituting Social Capital and Collective Action. In: R. O. Kehane & E. Ostrom (eds.). *Local Commons and Global Interdependence: Heterogeneity and Cooperation in Two Domains*. Sage Publications: London. Pp. 125-160.
37. Prietula M.J., Carley K.M., and Gasser L. (1998). *Simulating Organisations*. AAAI & MIT Press.
38. Rouchier J., Bousquet F., Le Page C., and Bonnefoy J.-L. (Forthcoming). Multi-Agent Modelling and Renewable Resource Issues: the Relevance of Shared Representations for Interacting Agents. *Proceedings of the Second Workshop on Multi-Agent Based Simulation (MABS 2000), Boston, 8-9th July, 2000.* To appear in Springer LNAI series, eds. S. Moss & P. Davidsson.
39. Russell S. and Norvig P. (1995). *Artificial Intelligence: a Modern Approach*. Prentice-Hall.
40. SDML software system. URL http://www.cpm.mmu.ac.uk/sdml/
41. SWARM software system. URL http://www.swarm.org/index.html
42. Walters C. J. (1986) *Adaptive Management of Renewable Resources.* Macmillan, New York, USA.
43. WCED (1987) *Our Common Future.* Oxford: Oxford University Press.

44. Weiss G. (ed.) (1999) *Multiagent Systems*. The MIT Press: Cambridge, Massachusetts and London, England.
45. Westervelt J. (2000) *Simulation Modeling for Watershed Management*. Springer-Verlag Telos.

Cooperating Physical Robots: A Lesson in Playing Robotic Soccer[*]

Bernhard Nebel

Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, Geb. 52
D-79110 Freiburg, Germany

Abstract. Having a robot that carries out a task for you is certainly of some help. Having a group of robots seems to be even better because in this case the task may be finished faster and more reliably. However, dealing with a group of robots can make some problems more difficult. In this paper we sketch some of the advantages and some problems that come up when dealing with groups of robots. In particular, we describe techniques as they have been developed and tested in the area of robotic soccer.

1 Introduction

Having a robot that carries out a task for you, e.g., cleaning the floor or fetching the mail, is certainly of some help. Having a group of robots seems to be even better because in this case the task may be finished faster and more reliably. Sometimes one even needs a group to get the task done. For instance, playing robotic soccer requires a team of robots.

In general, some problems can be more easily, more reliably, or faster solved by a group of robots. For example, distributing mail or messages to many targets can be done faster with a group of robots, as has been demonstrated by the team of SRI robots winning one of the robot competitions at AAAI'96 [13]. Also basic tasks such as *self-localization* can be more reliably solved by a group of robots. On the other hand, dealing with a group of robots can make some problems more difficult. For instance, path planning is easier for one robot than for a group of robots.

While this sounds all very plausible, it also raises the question why a scenario of a cooperating team of robots is interesting from a scientific point of view. A cooperating group of robots appears simply to be a special case of a cooperating group of agents. This is certainly true in the same way as is the statement that robots are "simply" special cases of agents. Mobile robots are special in a number of ways. For this reason, one has to deal with problems that do not arise with other agents, e.g., software agents. Firstly, there is the problem that a robot has to perceive and to act in a physical world. Secondly, sensing and acting is uncertain. Thirdly, connected with the two former points, communication between the robots might not be possible, be restricted to low bandwidth, or

[*] This work has been partially supported by *Deutsche Forschungsgemeinschaft* as part of DFG project Ne 623/3-1

possible only over restricted distances. Apart from that, however, groups of robots can be viewed as multi-agent systems.

One should note, however, that a group of robots may also be viewed as one centrally controlled multi-bodied robot. While such a viewpoint is indeed possible, there are a number of arguments against such a perspective. Firstly, this multi-bodied robot has a very large number of *degrees of freedom* making it computationally infeasible to control it. Secondly, we might be unable to communicate between the different parts of the robot or the communication bandwidth is very low. Thirdly, we might be unable to estimate a global system state because for some parts of the robot we do not know the state. Fourthly, failures of parts of the multi-bodied robot are much more naturally dealt with when one takes a multi-agent perspective.

In the rest of the paper, we will present some case studies of techniques developed in the context of multi-robot systems. In the next section, we have a look at *cooperative sensing*. In Section 3, we then turn to a particular form of *coordinated behavior*, namely, *cooperative motion planning*. In order to support cooperative behavior in a group, often *roles* are assigned to the group members. How this can be done for a group of robots in a highly dynamic environment will be studied in Section 4. Finally, in Section 5, we will discuss what is needed to build a successful robotic soccer team and in how far the techniques described in this paper can help.

2 Cooperative Sensing

If there is a group of robots that can communicate with each other, it seems natural that the robots share their observations with each other. In this manner they can compensate for sensor limitations that, for instance, restrict the range in which an object can be sensed. Furthermore, by combining estimates, robots may be able to narrow down their hypotheses or to correct their estimates.

As mentioned in the Introduction, all sensor measurements are uncertain. There is always some (normally distributed) noise and in addition there might be some systematic error one cannot anticipate. For example, when using the odometry – measuring how often a wheel has turned – there is a normally distributed measurement error and depending on the floor, there may be an additional systematic error. In particular carpets can can lead to systematic diversions that cannot be anticipated. Finally, there may also be a large displacement from time to time when the robot collides with an obstacle or with another robot. Worse yet, these errors accumulate leading to high uncertainty about the robot's position after a very short time.

For these reasons, other means are used to solve the so-called *self-localization* problem. Measurements from other sensors are used to correct the estimates derived from the odometry. The mathematical tool that is used to deal with this problem is often the *Kalman filter* [17], a method of fusing all measurements in order to arrive at optimal estimates. Intuitively, it involves computing a weighted average over sensor measurements, where sensors which are more accurate have a higher weight than those which are known to be less accurate.

Often, known positions of recognized landmarks are used in the self-localization process for correcting the position estimates. However, when one wants to explore an unknown territory, there are no known landmarks. With a group of robots that have initially known positions, it is possible to do something similar to landmark-based navigation, though. Some of the robots can be used as landmarks.

2.1 Cooperative Self-Localization

Rekleitis *et al.* [20] proposed a scheme for multi-robot exploration with a group of robots. In this approach it is assumed that the robots can track each other with reasonable reliability and accuracy as long the line of sight between them is free of obstacles. Under this assumption, one or more robots can move using one or more (temporary) immobile robots as landmarks. After a while the roles of the moving and immobile robots can be exchanged. Using such a method, the odometry error can be reduced dramatically [20].

In most applications, however, we already know the environment and "only" have to solve the self-localization problem. In this case it often happens that one robot can come up with multiple position hypotheses. If we now have a group of robots that are able to recognize other group members when they are close enough, it is possible that the robots narrow down the set of position hypotheses when they meet [8].

2.2 Cooperative Object Localization

Yet another scenario for multi-robot cooperative sensing is when we can assume that position and orientation of all robots are almost always accurate and reliably, but there is significant uncertainty and unreliability in sensing other objects. This situation occurs, for instance, in the robotic soccer context.

The players of the *CS Freiburg* team [11,18,23] use laser range finders in order to solve the self-localization problem [12], and for this reason can be assumed to know their own position very reliably. However, they are not very good in recognizing the ball and estimating its position on the field – which is done using a monocular vision camera.

There is a significant measurement error for estimating the distance to the ball, an error which increases with the distance between camera and ball. The angular error, on the other hand, is smaller and does not depend on the distance. Additionally, there is a restriction to the maximum distance over which the ball can be recognized, which is approximately 4–5 meters. Finally, the robots often enough recognize *false positives*, i.e., *phantom balls*. Ignoring the latter problem, one can again use a Kalman filter to fuse observations (with time stamps) from different robots in order to get estimates that a more accurate than any single measurement. In fact, this gives us a sort of stereo vision with a group of robots. Assuming that the angular error is much smaller than the distance error gives a triangulation effect as shown in Figure 1.

As already pointed out, sometimes the robots observe phantom balls. An example of such a situation is displayed in Figure 2. Two players see a ball close to the goal and another player sees a ball on the center line.

If we would now take the weighted average of the sensed ball positions, we would get a completely wrong estimate. For this reason, it seems preferable to exclude obviously wrong measurements. One way to do so would be to ignore measurements that are

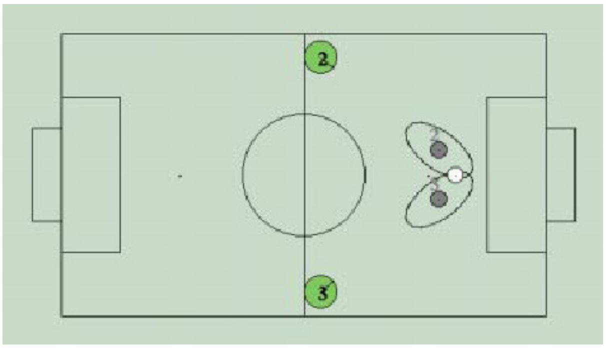


Fig. 1. The Kalman filter for integrating ball observations leads to triangulation. Grey discs denote position estimates for single robots, the ellipses around the grey discs denote measurement errors, and the white disc denotes the fused estimate.

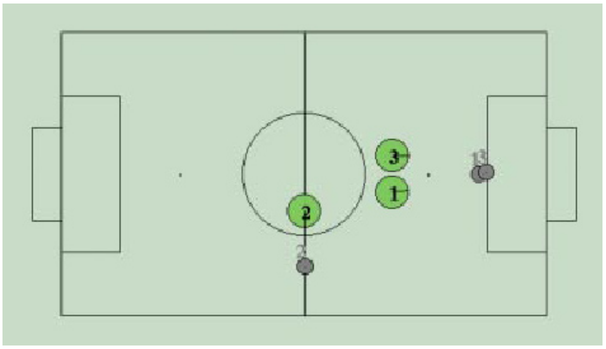


Fig. 2. Player 2 observes a phantom ball, which may lead to an incorrect estimate of the ball position.

implausible given the current estimate. This, however, could lead to a situation where a robot tracks a phantom ball and the other robots are all diverted from sensing the right ball because they believe the hallucinating robot.

The best (and most democratic way) to deal with such a situation is to believe in what the majority of robots sense. In the example depicted in Figure 2 we would rather believe the players 1 and 3 than player 2. One way to put such a *voting scheme* into effect is to use the so-called *Markov localization* approach [9] for the ball. In this approach one basically maintains (a discrete) probability distribution for the position probability of the object of interest. Usually, this is the robot itself. In our case, however, it is the ball. Each observation updates the position probability by increasing the probability at the location where the ball has been observed and lowers the probability where no observation has been made (using conditional observation probabilities and Bayes' rule). In addition, the position probability is flattened out for each time step to model the loss of certainty over time. Using such an approach, it appears more probable that the ball is around the locations where it has been observed by two robots than at the location where it has been observed only by one robot. Combining Markov localization as a plausibility filter with a Kalman filter, one gets a quite reliable and accurate global ball estimation mechanism [6].

While all these methods might not appear to be overly sophisticated, the real value of these approaches is that they are based on a solid theoretical basis and work in practice. Almost all of of these approaches, however, are still passive in the sense that they do not involve the interplay between sensing and acting, i.e., active sensing [3].

3 Cooperative Path and Motion Planning

Latombe starts his book [16] with the following remark:

> "This capability [motion planning] is eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world."

And what is true for single robots is, of course, also true for teams of robots.

The *basic motion planning problem* is usually stated [16] as the problem of moving a single rigid object – the *robot* – in an Euclidian (2- or 3-dimensional) space, the so-called *work space* from an initial position (and orientation) to a target position (and orientation). Of course, there can be *obstacles* in the workspace, which have to be avoided. The problem is usually solved by mapping the problem to the so-called *configuration space*. This space is generated by the *degrees of freedom* the robot has. In the 2-dimensional case, these degrees of freedom are (x, y, θ), i.e., the x and y coordinates of the robot position as well as its heading θ. In this 3-dimensional configuration space, the robot is just a point and we have to find a path from the start to the target configuration avoiding obstacles. In the special case that we have disk-shaped robots, the configuration space can be described by the x and y coordinates alone and so the configuration space is only 2-dimensional.

In the previous section, we have seen how sensing can lead to more accurate and reliable estimates if we have a group of robots. Furthermore, the additional computational costs are reasonable. In contrast to that, path and motion planning is computationally much more difficult if a group of robots is involved. This becomes obvious when one generalizes the configuration space planning method described above to a multi-robot system. In this case, for each robot 3 dimensions have to be added to the configuration space. Of course, this might be an indication that the configuration space approach is not appropriate. However, the multi-robot path planning problem is indeed inherently difficult. It is PSPACE-hard in the number of robots, as follows from results by Hopcroft *et al.* [14].

3.1 Cooperative Path Planning with Global Communication

If we assume that all robots can communicate with each other, the multi-robot path planning problem can be solved centrally, e.g., by using the configuration space approach sketched above. While this guarantees *optimality* and *completeness*, it is usually not efficient enough for even only a moderate number of robots.

Instead of a *centralized* approach, one can use *decoupled planning* [16]. In this approach, one plans first independent trajectories for all robots and then combines them, resolving conflicts when they arise. This reduces the complexity, but it also sacrifices optimality and completeness.

There are two decoupled planning methods that have been considered in the literature. First, there is the *prioritized planning* approach [7], which considers the multi-robot path planning problem as a sequence of path planning problems. One starts with the first robot and all the immobile obstacles. Then one adds another robot and plans a path avoiding all immobile robot and the moving robot from the first phase, and so on. The critical decision is what order one should use. Depending on this order, it is quite possible that no solution is found although there exists one.

The other decoupled approach is the so-called *path coordination* method [19], where the robots plan their paths independently (see Figure 3), and afterwards coordinate their

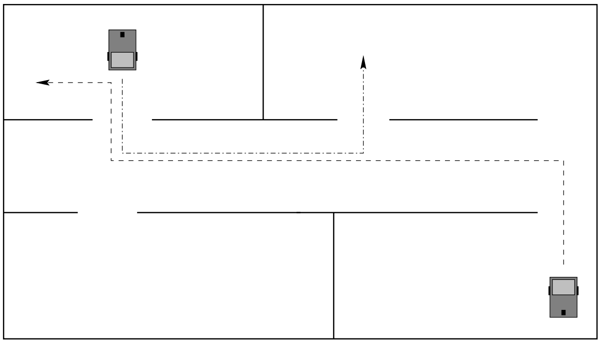


Fig. 3. Paths planned in a decoupled approach

movements without leaving their planned paths. As is obvious from this example, there is a chance that the two robots collide if they follow their planned paths without coordination. What we need here is a *collision-free schedule* the robots can follow. For two robots, this problem can be solved using so-called *coordination diagrams*, as the one shown in Figure 4. While this coordination diagram shows that with two robots the problem can be easily solved, it also gives a hint that the problem might become computationally difficult when the number of robots increase. The generalized coordination diagram would contain as many dimensions as there are robots. For this reason, often prioritization schemes are used [1].

Path coordination is, of course, even more restrictive than the prioritized planning approach and for this reason may find fewer solutions, which may be less cost-effective. This is indeed the case even in natural environments, as has been shown by Bennewitz and Burgard [2].

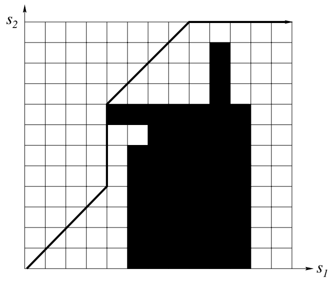


Fig. 4. Coordination diagram. The s_1 and s_2 axes represent the length robot 1 or robot 2, respectively, has already traveled on the respective planned path. Black cells represent collisions. The bold line shows a collision-free schedule.

3.2 Cooperative Path Planning with Only Local Communication

Although the decoupled path planning methods do not attempt to control the group of robots as one entity – and reduce the algorithmic complexity by that – these approaches still presuppose that there is a *central coordinator* and a *global communication network*. If one assumes that only *local communication* between pairs of physically close robots is possible, then the decoupled approaches do not work.

Similar to the decoupled approaches, we will assume that the paths are planned independently by each robot. However, instead of relying on a central component that deals with conflicts, we will now assume that only local coordination is possible [15].

If two robots are close to each other, they establish a coordination link, which means that they create a coordination diagram, which determines their schedule. It now can happen that one robot (A) has to wait for the other one (B). Unfortunately, it might happen that also robot B may have to wait for another robot C, which in turn waits for A, i.e., we have a *deadlock*. These deadlocks have, of course, to be detected and resolved. Resolution of a deadlock can here mean that one tries to find an alternative path in the coordination diagram or that a new path (segment) is planned [15]. The right tool to use here are *distributed algorithms* for deadlock detection and resolution [5].

4 Role Assignment in Dynamic Environments

The previous two sections have focused on problems such as sensing and path planning. In these cases, the solutions appeared to be very robotics specific and the overlap with multi-agent system seems to be minimal. However, there are, of course, other multi-robot problems that have a true multi-agent flavor.

One such problem is the *assignment of roles* to members of a group [21]. Such an assignment serves the purpose of associating a set of behavioral patterns with the agents in order to support the coordination between the agents. For instance, in soccer, we distinguish at least between the roles *goalie*, *defender*, and *forward player*. There may be additional roles such as *midfielder* and *supporter*. In general, we want to determine a one-to-one function from the set of agents A to the set of roles R. Sometimes, we may want also to consider different sets of roles, i.e., different *formations*. In soccer, for instance, we may want to deal with a 4-3-3 and a 3-3-4 formation and to switch between these formations.

A very simple way of dealing with this problem is to use a fixed assignment, as for example the *CS Freiburg* team did in 1998 [11]. Each robotic soccer player has a fixed role, which has an associated *home position* and an *area of competence*, as shown in Figure 5.

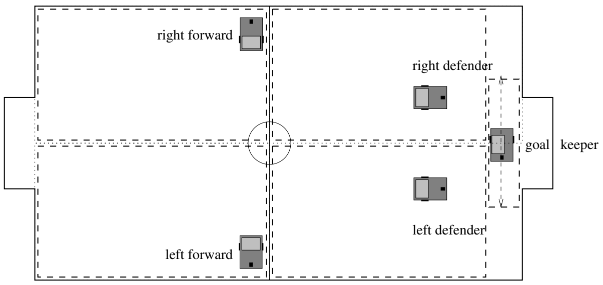


Fig. 5. Role assignment and areas of competence

When the ball moves into such an area of competence, the respective robot becomes active and tries to move the ball into the direction of the opponent goal – without leaving its area of competence. While this strategy does not appear to be optimal, it avoids the problem that a swarm of robot approaches the ball. In fact, only one robot of the team can be at the ball.

However, it is also clear that this delegation of duties has a number of severe problems. First of all, a defending robots can never run with the ball over the entire field and score a goal. They always have to pass the ball to a forward player. For this reason, very early on a "shouting" protocol was implemented that permits a robot with the ball to make a run to the opponent goal without being stopped by its own team members (see Figure 6).

A second disadvantage of the scheme described above is the disjoint decomposition of the field. It happened that the ball was in one of the competence areas, but the respective player was unable to go for the ball for some reason. Then no other player would come to help this player. This problem can be (and has been) solved by allowing overlaps of the competence areas and using the "shouting" protocol to avoid that two players block each other at the ball. Finally, there is the problem that once a robot breaks down, its role will not be filled by another robot on the field, even if the role is very important.

Although the *CS Freiburg* team became RoboCup world champion of the F2000 league in 1998, this was certainly not because its role assignment and coordination

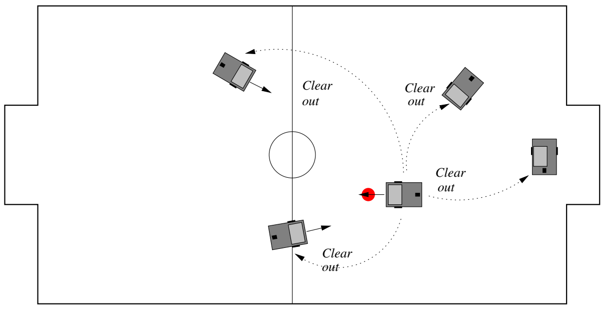


Fig. 6. Shouting in order to get a free run to the opponent goal

method were superior to that of the other teams. Indeed, there are a number of issues one has to address in order to build a flexible and robust team:

– role assignments should be changed dynamically to account for the current positioning and to support team reconfigurations after the break down or removal of individual team members; and
– flexible positioning that takes into account the entire situation on the field.

The latter point has been addressed by *CMUnited's* SPAR method [22], a method that tries to find the optimal position by using a linear programming method given the values for a number of important parameters such as ball position, position of team members, etc. Stone and Veloso [21] also addressed the issue of dynamic role re-assignments. However, this was approach was built on so-called *locker room agreements*, i.e., pre-built plans, and on filling more preferable roles if they are vacant.

A more flexible scheme for the dynamic assignment of roles has been proposed and used by the *ART Italy* team in 1999 [4]. They consider roles such as

– *active player*, the player which possesses the ball or goes to the ball;
– *supporter*, the player that moves parallel with the active player;
– *defender*, the player staying behind defending the goal.

Each agent can contribute some utility when filling a role. For instance, if a robot is already close to the defending position, it can contribute a high utility value when it fills the *defender* role. If it is close to the ball, it can contribute a high utility value if it fills the *active player* role. Each robot determines these utility values for each role and transmits the computed values to all other robots.

The roles for the field players are ordered by importance and assigned dynamically (in the importance order) to the player that can contribute most by filling the role according to the computed values. In fact, this assignment is done in a distributed manner, i.e. each robots decides on the basis of the received utility values which role to take. This can lead to situations, where a role is temporarily filled by two players. However, this does not happen very often and is resolved after a fraction of a second [4].

While this scheme appears to work very well, its efficiency seems to rely on ordering the roles by importance. A more general scheme would view the role assignment problem as an optimization problem, where we want to maximize the social welfare of the entire group. While this sounds like a combinatorial, i.e., a computationally difficult, problem, it is simply the problem of finding a *maximal weighted match* [10], which can be solved in polynomial time. In the *CS Freiburg* team, this more general scheme together with a communication protocol for changing roles in a consistent manner is used [23]. This is complemented by a variant of the SPAR method [22] to find the right position for each role.

5 Conclusions and Discussion: Playing Robotic Soccer

As should have become obvious from what has been said so far, robotic soccer is a rich source of inspiration for multi-robot and multi-agent research. It has already led to the development of a number of interesting new methods, and it is an attractive testbed for comparing different methods.

One of the interesting questions is in how far the multi-robot and multi-agent methods sketched in the previous sections could contribute to creating a competitive robotic soccer team. An answer depends, of course, on what *league* one has in mind. In the *simulation league*, multi-agent considerations are most probably of utmost importance. In the real robot leagues, however, the single agent capabilites are most important. If these capabilities (such as sensor interpretation or ball skills) are flawed, then even the best cooperation technique will not help. However, as mentioned in Section 2.2, cooperative sensing can compensate for the shortcomings of sensors. In fact, a number of goal scoring attempts by other teams could be stopped because of the global ball position estimation technique used by the *CS Freiburg* team.

Furthermore, with the increase of the single-agent capabilities over the years, cooperative behavior becomes more and more important. This development is also mirrored in the evolution of the *CS Freiburg* team. While in 1998 a robust and accurate sensor self-localization method together with simple ball skills and a very basic cooperation mechanism (see Section 4) was enough to win, team play proved to be one of the important skills in winning the RoboCup competition again in 2000 [23].

References

1. K. Azarm and G. Schmidt. A decentralized approach for the conflict-free motion of multiple mobile robots. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'96)*, pages 1667–1675. IEEE/RSJ, 1996.
2. M. Bennewitz and W. Burgard. An experimental comparison of path planning techniques for teams of mobile robots. In *Autonome Mobile Systeme*. Springer-Verlag, 2000.
3. W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, Aug. 1997. Morgan Kaufmann.
4. C. Castelpietra, L. Iocchi, M. Piaggio, A. Scalza, and A. Sgorbissa. Communication and coordination among heterogeneous mid-size players: ART99. In P. Stone, G. Kraetzschmar, and T. Balch, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Lecture Notes in Artificial Intelligence, Berlin, Heidelberg, New York, 2001. Springer-Verlag. To appear.

5. K. M. Chandy, J. Misra, and L. M. Haas. Distributed deadlock detection. *ACM Transactions on Computer Systems*, 1(2):144–156, May 1983.
6. M. Dietl, S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. 2001. Submitted paper.
7. M. A. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2(4):477–521, 1987.
8. D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Collaborative multi-robot localization. *Autonomous Robots*, 8(3), 2000.
9. D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.
10. Z. Galil. Efficient algorithms for finding maximum matchings in graphs. *ACM Computing Surveys*, 18:23–38, 1986.
11. J.-S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, and T. Weigel. The CS Freiburg team: Playing robotic soccer based on an explicit world model. *The AI Magazine*, 21(1):37–46, 2000.
12. J.-S. Gutmann, T. Weigel, and B. Nebel. Fast, accurate, and robust self-localization in polygonal environments. *Advanced Robotics Journal*, 2001. To appear.
13. D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige. Many robots make short work: Report of the SRI International mobile robot team. *The AI Magazine*, 18(1):55–64, 1997.
14. J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness for the 'warehousman's problem'. *International Journal of Robotics Research*, 3(4):76–88, 1984.
15. M. Jäger and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. 2001. Submitted paper.
16. J.-C. Latombe. *Robot Motion Planning*. Kluwer, Dordrecht, Holland, 1991.
17. P. S. Maybeck. The Kalman filter: An introduction to concepts. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*. Springer-Verlag, Berlin, Heidelberg, New York, 1990.
18. B. Nebel, J.-S. Gutmann, and W. Hatzack. The CS Freiburg '99 team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*. Springer-Verlag, Berlin, Heidelberg, New York, 2000.
19. P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'89)*, pages 484–489, 1989.
20. I. M. Rekleitis, G. Dudek, and E. E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1340–1345, Nagoya, Japan, Aug. 1997. Morgan Kaufmann.
21. P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.
22. P. Stone, M. Veloso, and P. Riley. The CMUnited-98 champion simulator team. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 61–76. Springer-Verlag, Berlin, Heidelberg, New York, 1999.
23. T. Weigel, W. Auerbach, M. Dietl, B. Dümler, J.-S. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski, and M. Thiel. CS Freiburg: Doing the right thing in a group. In P. Stone, G. Kraetzschmar, and T. Balch, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, New York, 2001. To appear.

A Multi-agent Study of Interethnic Cooperation

Vladimir Kvasnička and Jiři Pospíchal

Department of Mathematics
Slovak Technical University, 812 37 Bratislava, Slovakia
{kvasnic,pospich}@cvt.stuba.sk

Abstract. The purpose of this communication is to present an evolutionary study of cooperation between two ethnic groups. The used approach is reformulated in a form of evolutionary prisoner's dilemma method, where a population of strategies is evolved by applying simple reproduction process with a Darwin metaphor of natural selection (a probability of selection to the reproduction is proportional to a fitness). Our computer simulations show that an application of a principle of collective quilt does not lead to an emergence of an interethnic cooperation. When an administrator is introduced, then an emergence of interethnic cooperation may be observed. Furthermore, if the ethnic groups are of very different sizes, then the principle of collective guilt may be very devastating for smaller group so that intraethnic cooperation is destroyed. The second strategy of cooperation is called the *personal responsibility*, where agents that defected within interethnic interactions are punished inside of their ethnic groups. It means, unlikely to the principle of collective guilt, there exists only one type of punishment, loosely speaking, agents are punished "personally".

1 Introduction

The prisoner's dilemma game belongs to simple settings in which we can study different issues of cooperation in animal as well as human societies. The game has been extensively analyzed from many perspectives in the biological [18], social [1,2] and computational sciences [12]. Perhaps the most active area of research on the prisoner's dilemma concerns evolutionary versions of the game. The lower scoring strategies decrease in number, while this higher scoring strategies increase, and the process is repeated. Thus success in the evolutionary prisoner's dilemma depends more on doing well against other successful strategies, than with doing against a wide range strategies. The purpose of this communication is to use an evolutionary prisoner's dilemma to the study of cooperation between different ethnic groups, which is recently frequently discussed problem. Our concern to this very actual problem was initiated by recent seminal paper of Fearon and Laitin [6], where they used a standard iterated prisoner's dilemma theory to discuss different mechanisms of interethnic cooperation and conflicts. These authors very carefully discussed their results from the standpoint of equilibrium conditions, many useful theorems were proved. We modify and enlarge this approach in an evolutionary form, where a population of agents is considered. Each agent is endowed with a strategy of prisoner's dilemma game, and agents are either unpunished or punished. Moreover, the emerged strategies are discussed and interpreted in the framework of evolutionary stability theory [3,4,5,13,17], that belongs to the very effective theoretical tool for interpretation and better understanding of the evolutionarily emerged strategies.

For a typical setting of rewards/punishments of the prisoner dilemma a cooperation between two individuals is known to emerge only when the two interacting individuals meet repeatedly. They must remember each other past cooperative or deceptive interactions, and that memory must include at least one step in the history. Mutual cooperation for individuals does not usually arise, when two strangers meet just once, or they do not remember their mutual history. The history of mutual interaction can be however replaced by a general knowledge, whether the stranger was cooperative with others in the past. Such an approach was studied as image scoring together with its evolutionary stability [16]. We use a similar approach, only in an opposite sense, so that a noncooperating individual is marked for noncooperating, and this "penalty image" clings to him for some time period. This approach is very similar (only its negative) to the reputation in the image scoring, although it was produced independently and based on the previous work by Fearon and Laitin [6]. Such an approach can help to proliferate a cooperation. However, this approach is in our paper used only as a support in order to build up a theory of interethnic cooperation.

We study an approach, which can help us to model collective quilt and its possibility to support or hindrance an interethnic cooperation. We also study further approaches how to facilitate an interethnic cooperation. Our basic model is similar to that one of meeting of two strangers. This time the two interacting individuals, who do not have a memory of their past conduct, can recognize the type of ethnic group of the other individual. They do not know their personal history, neither their personal penalty or "image" obtained by encounters with other individuals. They only know, whether in the recent past some of the other group did not cooperate, which penalty then clings to the whole group.

This is an approach, which should model plausibly a basic situation in large urban areas, when individuals do not know themselves personally, and their encounters are unlikely to be repeated. However, the information "network" within the individual ethnic groups works well, so that an information about single interactions with other individuals from a different ethnic group can spread quickly within the ethnic populations.

We should emphasize, that we are not offering a "final theory" of ethnic conflicts that will cover all possible modes of ethnic peace and violence. Rather, we offer simple arguments and concepts to explain on a general level why an application of collective quilt in interethnic conflict is counterproductive. We will show that an application of this approach does not lead to an interethnic cooperation, even though the same principle works for intraethnic cooperation between individuals. Furthermore, this principle can be devastating for intraethnic cooperation within a smaller ethnic population interacting with another much larger ethnic population. The interethnic cooperation can be saved by introduction of an administrator. The administrator is supposed to watch over interethnic interactions and penalize defectors by taking away their reward. When this penalization is probable enough, the interethnic cooperation emerges. However, this police-like handling of a situation is not the most desired one. Therefore we looked for another principle how to support an emergence of cooperation.

In order to model situation, where the cooperation between two ethnic groups might emerge, we introduce a principle of a personal responsibility, where each noncooperating individual is "punished" by his/her own group. This approach should be a viable one, since an individual usually does not distinguish individuals from the other ethnic group, but they are able to distinguish each other within their ethnic group, and know by reputation the history of their members with other ethnic group interaction. If such individuals are excluded from cooperation within their own ethnic group, their type of behavior does not propagate. This method is implemented by a possibility, that an individual can marked for someone from the other ethnic group for defection. This "penalty image" can not be seen by other members from the punisher's group. During an interethnic interaction an individual does not any information about the opponent personal punishment or about the punishment of his group. However, the "penalty image" can be seen by other members the same ethnic group as is the punished one, and they are not able to distinguish, whether this punishment came for defection during interaction with another individual from his own ethnic group or from the other ethnic group.

The rest of this paper is organized as follows: Section 2 introduces basic principles of prisoner's dilemma game. Section 3 describes the basic principles of present multiagent approach and its application to intraethnic interactions. Section 4 describes an extension of the multiagent approach onto a study of interethnic interactions so that a principle of collective guilt is introduced. A modification of the principle of collective guilt so that an administrator is introduced is outlined in Section 5. Finally, Section 6 describes the so-called principle of personal responsibility that corresponds to another approach used for our simulations of interethnic cooperation.

2 Basic Principles of Prisoner's Dilemma Game

The prisoner's dilemma (PD) game [13,1] is a game between two players, each having the choice between two options: to cooperate (*C*) or to defect (*D*). The payoff matrix of this game is given by

$$A=\begin{pmatrix} R & S \\ T & P \end{pmatrix} \Leftrightarrow \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \tag{1}$$

where its entries are restricted by the following two inequalities

$$T>R>P>S,\quad 2R>T+S \tag{2}$$

The concrete numerical values in (1) are just specific examples, which were used in our calculations. If both players cooperate, they receive as payoff *R* (reward), which is assumed to be larger than the payoff *P* (punishment) received when both players defect. But if one player defects (plays *D*) while the other one cooperates (plays C), then the defector receives a payoff *T* (temptation) which is higher than *R*, whereas the cooperator receives a payoff *S* (sucker's payoff) which is lower than *P*. The second inequality (2) means that the total payoff for two players is larger if both cooperate than if one cooperates and another one defects. Formally, the payoff matrix may be understood as a payoff function

$$f:\{C,D\}^2 \rightarrow \{0,1,3,5\} \tag{3a}$$

determined as follows

$$f(C,C)=3,\ f(C,D)=0,\ f(D,C)=5, \text{ and } f(D,D)=1 \tag{3b}$$

The PD game is often successfully used in literature [9] for simulation of cooperation in biology, social sciences, economy, etc. In the present communication, the game PD will be extensively used in its evolutionary form [1,2] with resemblance to multiagent systems that belong to basic concepts of artificial life approaches. Our simulation results will be interpreted by making use of the very illustrative theory of evolutionary stability, originally elaborated with biological application of PD game by Maynard Smith [13], see also [17].

3 Intraethnic Cooperation

The purpose of this section is to outline an evolutionary PD to study an emergence of cooperation inside a single ethnic group. We use other classification scheme for strategies of single players than that one introduced by Axelrod [1,2]. He postulated that agents know short previous history of themselves and also all other agents. Then agent strategies depend on the previous moves of the opponent, whether (s)he cooperated or defected in interactions with agents in a few moves ago. In the present paper we use quite different approach how to express the history of interacting agents. After Fearon and Laitin [6], each agent is determined by the so-called punishment, which represents a label for other agents that the given agent has defected. The punishment plays a crucial role in a specification of interaction between two agents, its type depends whether none, one or both agents are punished.

Let us study a population P that corresponds to an ethnic group A

$$P = A = \{a_1, a_2, ..., a_m\} \tag{4}$$

As was already mentioned, each agent is specified by a punishment represented by a nonnegative integer

$$0 \leq p(a) \leq p_{max} \tag{5}$$

where p_{max} is a maximal punishment value. We say that an agent $a \in A$ is unpunished, if $p(a)=0$, in the opposite case, we say that he is punished, if $p(a)>0$. An interaction between two agents is fully determined by these punishments and strategy vectors. A strategy of an agent $a \in P$ is determined by the 4-dimensional vector

$$s(a) = (s_1, s_2, s_3, s_4) \in \{C,D\}^4 \tag{6}$$

Table 1. Specification of the strategy vector

No.	punishment of agent *a*	punishment of agent *b*	move of agent *a*	move of agent *b*
1	$p(a)=0$	$p(b)=0$	$s_1(a)$	$s_1(b)$
2	$p(a)=0$	$p(b)>0$	$s_2(a)$	$s_3(b)$
3	$p(a)>0$	$p(b)=0$	$s_3(a)$	$s_2(b)$
4	$p(a)>0$	$p(b)>0$	$s_4(a)$	$s_4(b)$

```
if (p(a)=0) and (p(b)=0) then
begin move_a:=s_1(a);move_b:=s_1(b);
      if move_a=D then p(a):=p_max;
      if move_b=D then p(b):=p_max;
end else
if (p(a)=0) and (p(b)>0) then
begin move_a:=s_2(a);move_b:=s_3(b);
      p(b):=p(b)-1;
end else
if (p(a)>0) and (p(b)=0) then
begin move_a:=s_3(a);move_b=s_2(b);
      p(a):=p(a)-1;
end else
begin move_a:=s_4(a);move_b:=s_4(b);
      p(a):=p(a)-1;
      p(b):=p(b)-1;
end;
```

Algorithm 1. Pseudopascal implementation of elementary interaction between two agents $a,b\in P$ that are specified by strategy vectors $\boldsymbol{s}(a)$, $\boldsymbol{s}(b)$ and punishments $p(a)$, $p(b)$. If both agents are unpunished, $p(a)=0$ and $p(b)=0$, then their moves are specified by the first entries of strategy vectors, $move_a=s_1(a)$ and $move_b:=s_1(b)$. In this case, defecting agent is punished, that is its punishment is set to the maximal punishment value, p_{max}. Further entries of strategy vectors are used (see Table 1) when any of agents is already punished. Moreover, the positive punishments are decreased.

Its single entries specify the types of interaction (C or D) for different cases specified in Table 1. For instance, if agents $a,b\in P$ are unpunished, $p(a)=p(b)=0$, according to the first entry of their strategis $\boldsymbol{s}(a)$ and $\boldsymbol{s}(b)$ agents play the following moves: $move_a=s_1(a)$ and $move_b=s_1(b)$. In general, actual moves of agents $a,b\in P$ with strategies $\boldsymbol{s}(a)$ and $\boldsymbol{s}(b)$ are determined in pseudopascal code by Algorithm 1. Formally, we may say that an agent $a\in P$ is represented by an ordered couple composed of the strategy and punishment

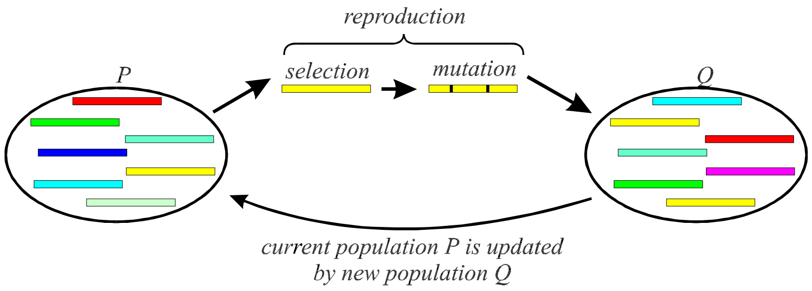


Fig. 1. A diagrammatic outline of the used evolutionary algorithm. From a current population *P* we quasirandomly select a strategy for a reproduction process, a probability of selection is proportional to the agent fitness. The reproduction process consists in a simple mutation of the selected strategy, i.e. its single entries (*C* or *D*) are sequentially mutated with a probability P_{mut} (small positive number, in all our simulation we put P_{mut}=0.001). The resulting new strategy is shifted to a new population *Q*, if its cardinality is equal to the cardinality of the current population, |*P*|=|*Q*|, then the current population is updated by the new population. The above-described process is repeated either a prescribed number of epochs or until the resulting population is composed almost entirely of the same (or similar) strategies.

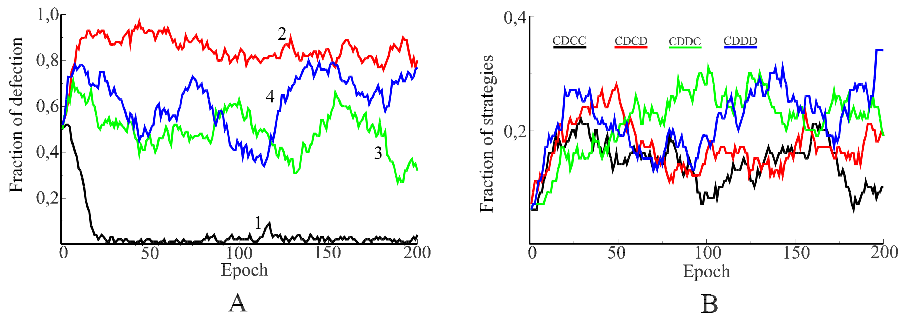


Fig. 2. Diagram A shows plots of defection fractions of single entries in strategy vectors from whole population. One can see that the first strategy entry very quickly tends to zero value, i.e. almost all strategies cooperate when in pairwise interactions both agents are unpunished. In similar way, the second strategy entry very quickly has achieved a value closely one. This observation means that almost all strategies defect in pairwise interactions if the current agent is unpunished while the other agent is punished. The third of fourth entries in population strategies strongly fluctuate from 0.4 to 0.8. It means that it is impossible to say whether an "average" population strategy cooperates or defects if the current agent is punished and the other agent is either punished or unpunished. To summarize the above discussed results, we can say that after 50 epochs the population of strategies is almost entirely composed of strategies represented by a mixture of strategies coded by a schema [10] (*CD##*), where hash symbols are substituted either by *C* or *D*. Diagram B demonstrates our conclusions from the previous diagram A, which means that in the course of evolution the population strategies can be formally represented by a schema (*CD##*), all others strategies that do not match the schema have vanishingly small fraction in the population.

$$a = (\boldsymbol{s}, p) \tag{7}$$

An interaction between two agents $a,b \in P$ is fully specified by their forms (7), i.e. by the corresponding strategies and punishments.

An evolution of population (composed of chromosomes that are specified by specification (7) consisting of agent's strategy and his punishment) is simulated by a simple version of evolutionary algorithm [10,8,7], where the crossover operation is omitted, see Fig. 1. This algorithm is based on proportional quasirandom selection of strategies for a reproduction process. In analogy to biology, we postulate that the strategy part of agent's specification corresponds to the agent's genotype, whereas the punishment part corresponds to an acquired property which is not inherited within the reproduction process.

Population of strategies is randomly initialized. Fitness of agents is calculated in such a way that a "tournament" is organized, where a prescribed number of times (N_{round}, in all our simulations we put $N_{round}=m \times m$, where m is population size) there are randomly selected two agents that perform one round of PD game, where the used moves are fully determined by their strategies and punishments (see Algorithm 1). The received payoffs are added to fitness of the corresponding agents.

Numerical results of our simulation calculations (performed for the population size m=1000 and maximal punishment $p_{\max}$=10) are presented in Fig. 2. We see that emerged strategies are examples of a schema (*CD##*), which (1) cooperates if both interacting agents are unpunished, and (2) defects if the first interaction agent is unpunished while the second interacting agent is punished (see the first two rows in Table 1). The emerged strategy resembles famous Axelrod's strategy [1,2] called the tit-for-tat strategy (TFT). One should be however aware, that tit-for-tat is not the only successful strategy in repeated prisoner's dilemma, see e.g. [14,15]. Our observations obtained from numerical results are summarized as follows:

(1) An evolution of population is controlled by two parameters: A critical frequency of agent interactions and a critical maximal punishment. The first parameter specifies a frequency of agent interactions, if it is smaller than its critical value, $0 \le N_{round} < N_{round}^{(critic)}$, then an evolution of population always produces a noncooperative strategy $\boldsymbol{s}$=(*DDDD*). On the other hand, if the frequency of agent interactions is greater or equal to a critical value, then the evolution is controlled by the maximal punishment. If the maximal punishment is smaller than a critical value, $0 \le p_{max} < p_{max}^{(critic)}$, than in similar way as above the evolution merely produces the noncooperative strategy $\boldsymbol{s}$=(*DDDD*); if the maximal punishment fulfils $p_{max} \ge p_{max}^{(critic)}$, then we get a class of strategies that are examples of the schema $\boldsymbol{s}'$=(*CD##*). This means that if the number of agent interactions is greater than a critical value, then the evolution of population is "bifurcated" onto two different branches. The first one corresponds to an emergence of entirely noncooperative strategy $\boldsymbol{s}$=(*DDDD*), whereas the second one is an emergence of cooperative strategies that are examples of the schema $\boldsymbol{s}'$=(*CD##*) (see Fig. 3).

(2) For an arbitrary value of the maximal punishment $p_{\max}$ the strategy ***s***=(*DDDD*) is evolutionary stable. A size of basin of attraction of the strategy ***s***=(*DDDD*) strongly depends on the values of the frequency of agent interactions and the maximal punishment p_{max}. For the smaller values of both parameters the basin of attraction of ***s***=(*DDDD*) is very broad and and may be composed of all-possible strategies. This means that in our simulations the resulting noncooperative strategy ***s***=(*DDDD*) should appear frequently as a result of population evolution. On the other hand, if the sizes of both control parameters are sufficiently great (i.e. $N_{round} \geq N_{round}^{(critic)}$ and $p_{max} \geq p_{max}^{(critic)}$), then the basin of attraction of ***s***=(*DDDD*) is considerably reduced and simulations begin to produce cooperative strategies that are examples of the schema ***s***'=(*CD##*). Loosely speaking, in order to achieve a cooperation within single ethnic group, the number of agent interactions and the maximal punishment should be of sufficiently great values. In other words it means that if interactions of agents are relatively infrequent and the punishment of defection is not sufficient, then a cooperation strategy could not emerge as an evolutionary result, all agents will mainly defect in interactions between them.

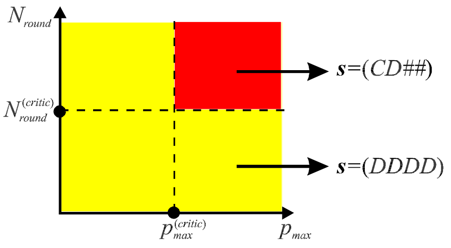


Fig. 3. Diagrammatic representation of basins of attraction in a dependence on the maximal punishment and the number of interactions for one evolutionary epoch. The lightly shaded area corresponds to values of parameters that produce the noncooperative strategy ***s***=(*DDDD*) , whereas the darkly shaded square area corresponds to values of parameters (both a greater or equal to their critical values) that produce cooperative strategies that are examples of the scheme ***s***=(*CD##*).

In our recent publication [11] we used a formalism of evolutionary stable strategies that was initially introduced by Maynard Smith [13] followed by [3,4,5,9,17] in the framework of his biologically oriented evolutionary game theory studies. We have enlarged the original Maynard Smith theory to the case when strategies are punished. Let us assume that a population is composed mostly of reference strategies s_0 and in a smaller extent of intruder strategies s_1, where both strategies may be either punished or unpunished. Loosely speaking, a strategy is called *evolutionarily stable* if in the course of forthcoming evolution intruder strategies extinct. Such an intuitive formulation, though acceptable from the standpoint of stability conditions of general dynamic systems [9], should be more deeply specified by a terminology of evolutionary methods. All the substantial computational results were checked and interpreted analytically within the theory of evolutionary stable strategies. Moreover, this theoretical approach offers mechanisms of simple scenarios explaining why some particular strategies are stable or not.

4 Principle of Collective Guilt and Interethnic Interactions

The purpose of this Section is to use the punishment approach from Section 3 to a study of interethnic cooperation. A guiding rule of this study will be the so-called principle of collective guilt, based on an assumption that the whole ethnic group is responsible for a defection done by its member in interethnic interactions.

Let us assume that a population P is composed of two ethnic groups A and B

$$P = A \cup B \left(A \cap B = \emptyset \right) \tag{8a}$$

$$A = \{a_1, a_2, ..., a_m\} \text{and} B = \{b_1, b_2, ..., b_n\} \tag{8b}$$

Between agents of the population P there exist three types of interactions:

(1) intraethnic interactions within the ethnic group A,
(2) intraethnic interactions within the ethic group B, and finally,
(3) interethnic interactions between agents from A and B.

After Fearon and Laitin [6] we postulate that a frequency of interethnic interaction is much smaller than frequencies of intraethnic interactions in A or B. This requirement will be simply achieved in our calculations so that we introduce a probability P_{inter} of interethnic interactions within a class of all possible interaction between agents from the whole population P. A value of this probability is set P_{inter}=0.1, i.e. only 10% fraction of all interactions is of the interethnic character.

Second basic requirement of our simulations is an assumption of the so-called information asymmetry, cf. ref. [6]. It is supposed that agents can identify others from the same ethnic group and that they know one another's history of play. On the other hand, agents in interethnic interactions know only that they are paired with someone from the other ethnic group, they cannot observe either the individual identity or personal history of inter-group agents. This requirement is not intended to characterize every type of interethnic interaction, e.g. a durable long-term relationship between two traders. Rather, it corresponds to a particular class of interactions that are plagued by a difficult problem for interethnic cooperation due to information asymmetry and relative infrequency. We have to emphasize that the principle of collective guilt is closely related to the above assumption of information asymmetry. A lack of information about agents from other ethnic group causes, for instance, that the principle of collective guilt is used as a guiding rule for punishment of interethnic defections.

Let each agent of $a \in A$ and/or $b \in B$ is specified by the punishment

$$0 \leq p(a) \leq p_{max} \tag{9a}$$

$$0 \leq p(b) \leq p_{max} \tag{9b}$$

All-ethnic punishments are determined by

$$0 \le p(A) \le p_{max} \tag{10a}$$

$$0 \le p(B) \le p_{max} \tag{10b}$$

these punishments will play an important role in interethnic interactions as a manifestation of our assumption of information asymmetry. Since each agent from the interethnic pairing knows nothing about his opponent, this lack of information is substituted by much weaker requirement, the interacting agents know only whether ethnic group of their opponents is punished or not.

Table 2. Specification of the last four entries in the strategy vector

No.	punishment of the group A	punishment of the group B	move of agent $a\in A$	move of agent $b\in B$
5	$p(A)=0$	$p(B)=0$	$s_5(a)$	$s_5(b)$
6	$p(A)=0$	$p(B)>0$	$s_6(a)$	$s_7(b)$
7	$p(A)>0$	$p(B)=0$	$s_7(a)$	$s_6(b)$
8	$p(A)>0$	$p(B)>0$	$s_8(a)$	$s_8(b)$

A strategy of an agent $a\in A$ is described by an 8-dimensional vector

$$\boldsymbol{s}(a) = (s_1, s_2, ..., s_8) \in \{C, D\}^8 \tag{11}$$

The first four components of this strategy vector determine an intraethnic interaction (see Table 1), the next four components determine an interethnic interaction of the agent $a\in A$ with another agent $b\in B$, see Table 2. The evolution of population (see Fig. 1) is done independently for both ethnic groups A and B. Ethnic groups are relatively independent in the course of evolution, they interact only when fitness of agents is calculated, which has a part originated from the interethnic interactions. It means that we may speak about a coevolution of strategies from the ethnic groups A and B.

In similar way as in Section 3, fitness of agents is calculated by a "tournament", where a prescribed number of times (N_{round}, in all our simulations we put $N_{round}=m\times m+n\times n+m\times n$, where m and n are sizes of ethnic group A and B, respectively) there are randomly selected two agents that perform one round of PD game, the used moves are fully determined by their strategies and punishments (see Algorithms 1 and 2). The interethnic interactions are performed in the tournament with the probability P_{inte} , while the intraethnic interactions are performed with the probability $1-P_{inte}$.

Numerical results of our simulation calculations (performed for the group sizes $m=n=500$, maximal punishment $p_{\max}=10$, $P_{inte}=0.1$) are presented in Figs. 4 and 5. From the results one can conclude that evolution of strategies that are based on the principle of collective guilt do not produce an evolutionary stable strategy with build-in cooperation between pairs of agents taken from different unpunished ethnic groups, see Fig. 5. Though the resulting evolutionarily stable strategy is cooperating within intraethnic interaction of two agents that are unpunished, the main conclusion from our calculations is that the resulting strategy is not cooperating for interethnic interaction of two agents when both ethnic groups are unpunished simultaneously. The observations obtained from numerical results are summarized as follows:

(1) If we use for our simulation calculation the same parameters as those used in our simulations of cooperation within single ethnic group, then the resulting strategies for interethnic interaction are examples of a schema $\boldsymbol{s}_0$=(CD##DDDD). It means that within ethnic group there exists cooperation while for interethnic interactions the strategy offers only defections. This important observation was also verified by a model calculation, where the initial population is entirely composed of strategy $\boldsymbol{s}_1$=(CDDDCDDD), see Fig. 5. The strategy cooperative for interethnic interactions was quickly substituted by strategies that are examples of the interethnically-noncooperating schema $\boldsymbol{s}_0$=(CD##DDDD). In other words, the cooperative strategy $\boldsymbol{s}_1$=(CDDDCDDD) is not evolutionary stable, that is the principle of collective guilt does not lead to a cooperation between different ethnic groups.

(2) Loosely speaking, the principle of collective guilt is unable to guarantee an emergence of cooperation between different ethnic groups. Collective punishment is not sufficiently specific, interethnic defections is spread out through the whole ethnic group and therefore it could not be used as a driving force for an emergence of interethnic cooperation.

```
if (p(A)=0) and (p(B)=0) then
begin move_a:=s_5(a);move_b:=s_5(b);
      if move_a=D then p(A):=p_max;
      if move_b=D then p(B):=p_max;
end else
if (p(A)=0) and (p(B)>0) then
begin move_a:=s_6(a);move_b:=s_7(b);
      p(B):=p(B)-1;
end else
if (p(A)>0) and (p(B)=0) then
begin move_a:=s_7(a);move_b:=s_6(b);
      p(A):=p(A)-1;
end else
begin move_a:=s_8(a);move_b:=s_8(b);
      p(A):=p(A)-1;p(B):=p(B)-1;
end;
```

Algorithm 2. Pseudopascal implementation of an elementary interaction between two agents from different ethnic groups. Let agents $a \in A$ and $b \in B$ be determined by strategy vectors $\boldsymbol{s}(a)$, $\boldsymbol{s}(b)$ and by group punishments $p(A)$, $p(B)$. If both ethnic groups are unpunished, $p(A)=p(B)=0$, then their moves are specified by the fifth entry of their strategy vectors, $move_a=s_5(a)$, $move_b=s_5(b)$. In this case the ethnic group of defecting agent is punished, i.e. the group punishment is set to the maximal punishment value $p_{\max}$. Further entries of strategy vectors specify possible situation if either group A or B is punished, see Table 2.

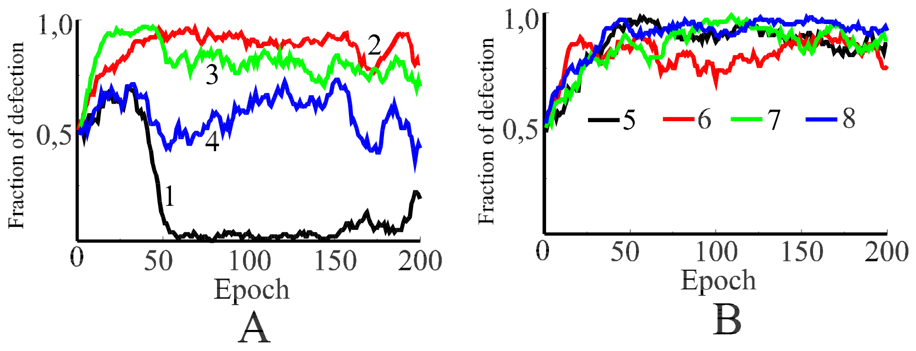


Fig. 4. Diagrams A and B show plot of defection fraction corresponding to the 1-4 entries and 5-8 entries, respectively, of population strategy vectors. We see that the resulting strategies can be represented by a scheme ***s***=(*CD##DDDD*). We have to note that a definite value (*C* or *D*) of the fourth entry does not follow uniquely from our calculations, in some stages of evolutionary history fractions of defection in the fourth position are about 1/2, i.e. *C* or *D* have the same probability of appearing. This may be explained by a low probability of interaction of two punished agents, so that the mutation can change the defection fraction randomly. The resulting strategy means that agents cooperate within intraethnic interaction when both agent are unpunished, but agents always defect for interethnic interactions independently on group punishments.

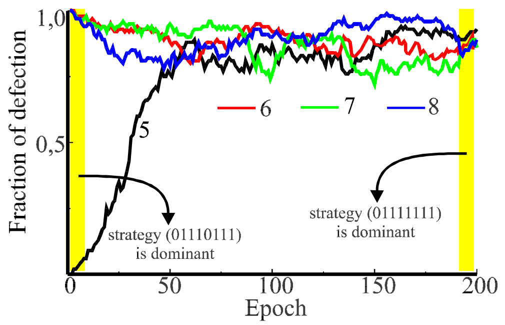


Fig. 5. Diagrammatic presentation of our model calculation, where the population *P* was initialized entirely by (*CDDDCDDD*) strategies. This means that both interacting agents that are unpunished (irrespectively of the interaction type) cooperate, and in all other cases they defect. We see from the plots that this "stationary" strategy is not evolutionary stable, in the course of evolutionary process it was spontaneously transformed to the strategy (*CDDDDDDD*). Consequently, we may say that the strategy (*CDDDDDDD*) (see Fig. 4) is evolutionary stable.

In the last part of this Section we will study separately a special case of interethnic interactions, when the sizes of ethnic group are very different, $|A|>>|B|$, i.e. a number of members of *A* is much greater than a number of *B*. For example, if $|A|=1000$ and

$|B|$=100, and there are 50 interethnic interactions per one evolutionary epoch, then a probability for an $a \in A$ of an interethnic interaction encounters per epoch is p_A=0.05, while it is p_B=0.5 for an agent $b \in B$. This implies that for a smaller ethnic group B interethnic interactions might be of considerably great impact than for a greater group B. In order to verify this hypothesis we repeated our computer simulations for ethnic groups $|A|$=1000 and $|B|$=100, where we put $N_{round}=10^6$, the maximal punishment is $p_{\max}$=10, and the probability of an interethnic interaction is P_{inte}=0.1 (i.e. only 10% interactions from N_{round} are of interethnic character). Diagrams in Fig. 6 represent the obtained numerical results. We see that the principle of collective guilt for smaller ethnic group is devastating, it causes an extinction of intraethnic cooperation. Thus, if the agents of smaller group tend to have relatively high interethnic interaction rates, then for such a group intraethnic cooperation can be temporarily destroyed as a consequence of interethnic interactions that are of dominant character with respect to intraethnic interactions.

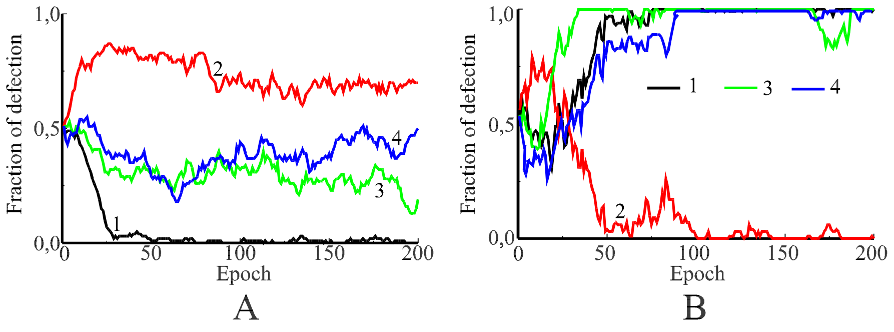


Fig. 6. Plots of fraction of defections for the first four entries of strategies for ethnic groups *A* and *B* that are of different size, $|A|>>|B|$. Diagram A (ethnic group *A*) shows that the emerged strategy is intraethnically cooperating, that is when both agent are unpunished then the agent moves *C* (cooperation). On the other hand, diagram B (smaller ethnic group *B*) shows that the emerged strategy is intraethnically defecting.

In the Fig. 6, plot A shows, that for the group A the first entry in average approaches 0, which means cooperation with unpunished agents of A and the second entry approaches 1, which means defection with punished agents of A. The lines for other two entries move stochastically around 0.5, since the chance of interaction of two punished agents of A is so small, that random mutations destroy any regularity. The other plot B for an intraethnic cooperation shows destruction of cooperation for unpunished agents, and surprisingly a cooperation with punished agents. This result is however of transient type, the strategy in further generation can switch back and forth to the strategy (*CDDDDDDD*). This instability is caused by a genetic drift, since the population B is ten times smaller than A, and the other part of chromosome for interethnic cooperation influences B much more than A, because interethnic interactions occur for B ten times more often.

5 Principle of Collective Guilt with an Administrator

In the previous Section 4 we have observed that the principle of collective guilt applied to interethnic interactions does not produce interethnic cooperation. The purpose of this Section is to study simple modifications of this principle so that an administrator, which punishes a defection, is introduced.

A modification of the principle of collective guilt by introduction of an administrator was initially discussed by Fearon and Laitin [6] as an alternative approach that is able to ensure interethnic cooperation in the model based on the principle of collective guilt. An administrator is introduced that stochastically punishes those agents that defected within interethnic interactions (both ethnic groups are unpunished), see Algorithm 3. According to our basic assumption of the information asymmetry, it is plausible to postulate that the administrator lacks detailed knowledge about just interacting agents from different ethnic groups. Suppose than an $a\in A$ agent defects a $b\in B$ agent. Then the administrator in a style of *deus ex machina* (e.g. a member of the Royal Canadian Mounted Police) stochastically penalizes those agents that defected in interethnic interaction, when both ethnic groups were unpunished.

Our numerical results show that such a simple modification of our model (we set penalization probability to P_{penal}=0.5, i.e. roughly 50% of interethnic defections are penalized) leads to an emergence of an interethnic cooperation, see Fig. 7. The penalization probability has a threshold value, below the threshold interethnic cooperation does not arise. It means that there exists a critical value for a frequency of moderator penalization acts, an emergence of interethnic cooperation requires a relatively high frequency of the moderator penalizations if interethnic defection occurs.

```
if (p(A)=0) and (p(B)=0) then
begin move_a:=s_5(a);move_b:=s_5(b);
      if move_a=D then p(A):=p_max;
      if move_b=D then p(B):=p_max;
      payoff_a:=f(move_a,move_b);
      payoff_b:=f(move_b,move_a);
      if(payoff_a=5)and (random<P_penal) then payoff_a:=0;
      if(payoff_b=5)and (random<P_penal) then payoff_b:=0;
end else .....
```

Algorithm 3. A simple modification of the Algorithm 2. If both ethnic groups are unpunished and if one agent defects and the other one cooperates, then the defector is immediately penalized with a probability P_{penal} by the administrator so that its current received payoff is cancelled.

A stability study of interethnic interactions based on the collective guilt with an administrator can be simply done by an approach outlined in a recent publication [11]. The main result of this evolutionary study is, that the probability P_{penal} should be greater than a critical value.

$$P_{penal} > \frac{2}{5} \qquad (12)$$

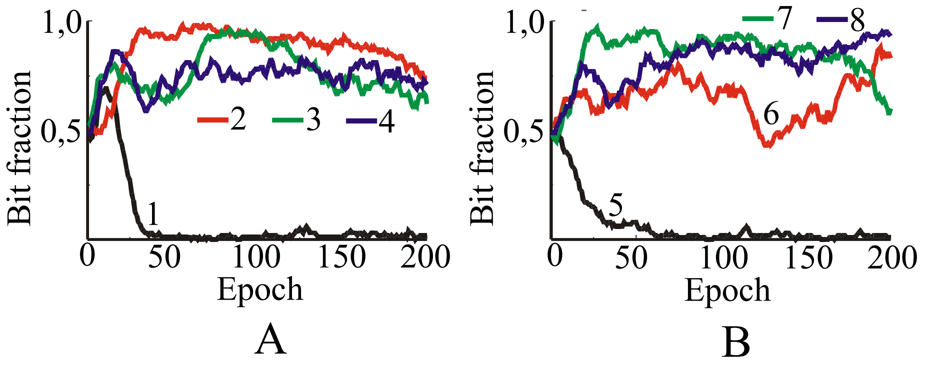


Fig. 7. Plots of single bit fractions for intraethnic (A) and interethnic (B) interactions, when an administrator was introduced, specified by the probability of penalization P_{penal}=0.5. Both diagrams, that corresponds to the ethnic group A (for the other group B we get similar plots), demonstrate that intraethnic as well as interethnic cooperation has emerged if the principle of collective guilt is "moderated" by an administrator that penalizes interethnic defection.

It means that interethnic cooperation emerges for a sufficiently great "reliability" of administrator check activities. If an administrator check activity is not very reliable, then the introduction of the administrator does not cause an emergence of interethnic cooperation. Hence, providing that a "reliability" of administrator check activity is sufficiently great, the reference strategy $\boldsymbol{s}_0$=(*CDDDCDDD*) is evolutionarily stable with respect to an intruder strategy $\boldsymbol{s}_1$=(*CDDDDDDD*), which was to be proved. We have to emphasize, that the result (12) means that for a penalization probability P_{penal} there exists a threshold value, if this probability is greater than a critical value, then there exists an emergence of cooperation between different ethnic groups. Furthermore, decreasing the value P_{penal} will cause a decrease of a frequency of interethnic cooperation.

6 Principle of Personal Responsibility and Interethnic Interactions

In Section 4 we have elaborated a model of interethnic interactions based on the principle of collective guilt. There was demonstrated that an application of the principle does not produce an emergence of the interethnic cooperation between ethnic groups *A* and *B*. The interethnically cooperative strategy (represented by a scheme (C###C###)) may be formally considered only as a stationary one, spontaneously emerging (by mutations in reproduction process) interethnically noncooperating strategies (represented by a schema (C###D###)) overcome the interethnically cooperating strategy. Moreover, for ethnical groups of unequal sizes, the principle of collective guilt might be very devastating for smaller ethnic group so that intraethnic cooperation is disintegrated. These and other arguments lead us to consider another model of interethnic interaction based on the principle of personal responsibility.

Table 3. Specification the last two entries in the strategy vector

No.	punishment of A	move of $a\in A$
5	$p(A)=0$	$s_5(a)$
6	$p(A)>0$	$s_6(a)$

Let us substitute the principle of collective guilt by a principle of personal responsibility, where agents that defect within interethnic interactions are punished inside of their ethnic groups. It means that contrary to the principle of collective guilt there exists only one type of punishment, in particular agents are punished "personally". An agent that did not cooperate within a single act of interethnic interaction is punished inside of its ethnic group, so that the agent is considered as being punished by other members of the same group.

A strategy of an agent $a\in A$ is described by a 6-dimensional vector

$$\boldsymbol{s}(a)=(s_1,s_2,....,s_8)\in\{C,D\}^8 \tag{13}$$

The first four component of this strategy vector determine an intraethnic interaction (see Table 1), the next two component determine an interethnic interaction of the agent $a\in A$, see Table 3. A move of the first agent in the interethnic interactions is fully determined by its punishment (a punishment of the second agent-opponent does not affect the first agent). This interpretation of interethnic interactions is a consequence of our initial assumption of information asymmetry, where it is supposed that agents can identify others from the same ethnic group and that they know one another's history of play. On the other hand, agents in interethnic interactions know only that they are paired with someone from the other ethnic group. An interethnic interaction of two agents $a\in A$ and $b\in B$ is specified in pseudopascal in Algorithm 4.

An application of the principle of personal responsibility leads to an emergence of cooperation within intraethnic as well as interethnic interactions. Diagrams in Fig. 8 illustrate our typical results obtained the group sizes m=n=500, maximal punishment $p_{\max}$=10, and probability of interethnic interaction P_{inte}=0.1. Our observations from numerical calculations are very similar to those ones summarized at the end of Section 3 dedicated to intraethnic cooperations. In particular, we have observed that two types of strategies are evolutionarily stable. First, the totally defecting strategy $\boldsymbol{s}$=($DDDDDD$) is evolutionarily stable, the size of basis of its attractiveness considerably depends on the size of maximal punishment. When the maximal punishment is increased, the basin of attractiveness tends to smaller sizes. Second, for a sufficiently great maximal punishment p_{max} the cooperating strategies that are examples of the schema $\boldsymbol{s}$=(C####CD) became evolutionarily stable, when the population of strategies is randomly initialized.

The above mentioned observations are fully interpretable in the framework of theoretical approach outlined in our recent publication [11]. There exists some obstacle in a straightforward application of the formalism of evolutionary stability, in the present theory of simultaneous intra- and interethnic interactions the punishment of strategies is of the same type regardless of intra- or interethnic character of interaction. This means that we cannot theoretically study an evolutionary stability of strategies separately for inter- as well as intraethnic interactions. Nevertheless, the theory is also straightforwardly applied to the study of evolutionary stability of strategies evolved in a population composed of two ethic groups with interactions that are governed by the principle of personal responsibility. What is slightly upsetting here, the resulting formulae are lengthy and their derivation is clumsy. These are main reasons why we turn our attention to a simple "informal" discussion of the obtained results.

```
if p(a)=0 then
begin move_a:=s_5(a);
      if move_a=D then p(a):=p_max;
end else
begin move_a:=s_6(a);
      p(a):=p(a)-1;
end;
if p(b)=0 then
begin move_b:=s_5(b);
      if move_b=D then p(b):=p_max;
end else
begin move_b:=s_6(b);
      p(b):=p(b)-1;
end;
```

Algorithm 4. Pseudopascal specification of interethnic interaction between two agents $a\in A$ and $b\in B$ that are determined by strategy vectors $\boldsymbol{s}(a)$, $\boldsymbol{s}(b)$ and punishments $p(a)$, $p(b)$. If an unpunished agent $a\in A$ ($b\in B$) defects, then it is punished by $p(a)=p_{\max}$ ($p(b)=p_{\max}$).

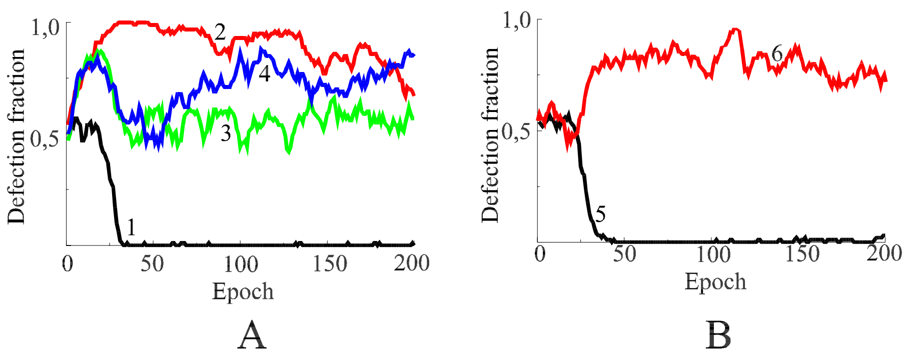


Fig. 8. Diagrams A shows plots of the first four strategy-vector entries that are responsible for the interethnic cooperation. We see that the first entry s_1 quickly tends to zero, i.e. an emerging strategy cooperates for intraethnic interactions. Diagram B shows plots of the fifth and sixth strategy-vector entries. The fifth entry s_5 quickly dents to zero, i.e. the emerged strategy cooperates for interethnic interaction. The emerged strategies may be represented formally by a scheme (**C**### *CD*).

Let us consider a strategy $\boldsymbol{s}$ that is represented as a "direct sum" of an intraethnic strategy and an interethnic strategy $\boldsymbol{s}_{inter}$

$$\boldsymbol{s} = \boldsymbol{s}_{intra} \oplus \boldsymbol{s}_{inter} \tag{14}$$

where the strategy $\boldsymbol{s}_{intra}$ corresponds to the first four components of the "whole" strategy $\boldsymbol{s}$, whereas the strategy $\boldsymbol{s}_{intra}$ correspond to the fifth and sixth entries of $\boldsymbol{s}$. Let us define two agents a and a' that share the same strategy $\boldsymbol{s}$ but have different punishments

$$a = (\boldsymbol{s}, p = 0) \quad \text{and} \quad a' = (\boldsymbol{s}, p' > 0). \tag{15}$$

It means that both agents correspond to the same strategy specified by (14) but to different punishments, the first agent is unpunished whereas the second agent is punished. Finally, let us postulate that the strategy $\boldsymbol{s}_{intra}$ is cooperative within ethnic group, e.g. $\boldsymbol{s}_{intra}=(CDDD)$, while its counterpart $\boldsymbol{s}_{inter}$ is not cooperative, e.g. $\boldsymbol{s}_{inter}=(DD)$. Let the agent $a=(\boldsymbol{s},p=0)$ takes part in an interethnic interaction, according to noncooperative interethnic part, as a result of this interaction a new agent $a'=(\boldsymbol{s},p'=p_{max})$ is formed. Since the interethnic interactions are relatively rare, we may postulate that in the forthcoming history (within one evolutionary epoch) the agent a' takes part only in intraethnic interactions. According to its punishment, the resulting fitness should be smaller than a fitness of a hypothetical unpunished agent $\tilde{a} = (\boldsymbol{s} = \boldsymbol{s}_{intra} \oplus \tilde{\boldsymbol{s}}_{inter}, \tilde{p} = 0)$, with an interethnic "cooperative" part $\tilde{\boldsymbol{s}}_{inter}$, e.g. $\tilde{\boldsymbol{s}}_{inter} = (CD)$

$$fitness(\tilde{a}) > fitness(a) \tag{16}$$

It means that those agents that have noncooperative interethnic parts of strategies became extinct as a consequence of the fact that punishment is not distinguished according to its origin from intra- or interethnic interactions. Agents with noncooperative intraethnic parts of strategies became extinct from the population according to the mechanism of intraethnic interactions. On the other hand, extinction of agents with noncooperative interethnic parts of strategies is shifted to intraethnic interactions. Summarizing, we see that for an emergence of interethnic cooperation in a population endowed by the principle of personal responsibility plays important role a fact that punishments originated from interethnic defections is transferred to intraethnic interactions, where strategies that were defecting within interethnic interactions gather smaller fitness a therefore are forced to extinction.

7 Summary

Evolutionary PD is used to study an emergence of intra- and interethnic cooperation. In our computer simulations each population agent and the whole ethnic group is specified by a nonnegative integer that corresponds to the so-called punishment. This punishment parameter of agents and the whole ethnic group is used in specification of agent strategies when they take part in pairwise interactions. Actual performance of a single PD strategy game depends on the strategies of both agents and also their punishments.

We have studied a population of strategies that is composed of one ethic group. We have demonstrated that within single ethnic group an emergence of cooperative strategy depends on the size of maximal punishment and the number of interethnic interaction events per evolutionary epoch. If both of these are sufficiently small, then cooperative strategy ***s***=(*CDDD*) (an analogue of Axelrod's Tit-For-Tat cooperative strategy [1]) does not emerge, just to the contrary, noncooperative strategy ***s***=(*DDDD*) emerges. This noncooperative strategy is persisting also for higher values of the maximal punishment and increased number of interethnic interaction events, although a basis of its attractiveness is considerably smaller than for their smaller values. A new feature of these higher values is that for randomly initiated populations of strategies there exist an emergence of the cooperative strategy ***s***=(*CDDD*). These results are simply explained by an application of an evolutionary stability theory.

The above simple model for simulation of emergence of cooperative strategies within one ethic group can be simply extended so that it will cover also interethnic interactions (i.e. population is composed of two ethnic groups). We used two different approaches to extend the theory of intraethnic interactions.

First, the principle of collective guilt is based on an assumption than whole ethnic group is punished when an agent from this group defected within interethnic interaction. it means, that a responsibility for individual defection is spread out through the whole ethnic group. We demonstrated that this model of interethnic interaction does not lead to an emergence of interethnic cooperations. Only such strategies emerged that are intraethnically cooperative but interethnically defecting. This unpleasant feature of the principle of collective guilt can be removed if we introduce for both ethnic group an administrator (*deus ex machina*), that stochastically penalizes those agent that defected within interethnic interactions. An introduction of the administrator causes an emergence of interethnical cooperation, but this positive fact has to be paid by the relatively high frequency of administrator penalizations acts. If this frequency is relatively low, than administrator penalization does not induce the emergence of cooperation between different ethnic groups. Furthermore, if a population is composed of ethnic groups of substantially different sizes, then the principle collective guilt may be fatally devastating for a smaller ethnic group, so that in this group intraethnic cooperation is temporarily vanished and becomes erratic. This interesting observation can be simply explained in the framework of our evolutionary stability considerations in such a way that agents of smaller group tend to have relatively high interethnic interaction rates. For such a group a fitness is greatly influenced by an interethnic part of a strategy, which can act as a “noise” for evaluation of an intraethnic part of strategy. This “noise” together with the fact, that a smaller population is more susceptible to a genetic drift, causes long periods of noncooperative strategies dominance in intraethnic interactions.

Second, the principle of a personal responsibility is a natural extension of our theory of intraethnic interactions, agents that defect within interethnic interactions are punished within their ethnic groups. It means, contrary to the principle of collective guilt, there exists only one type of punishment, in particular agents are punished "personally". An agent that did not cooperate in a single act of interethnic interaction is punished inside of its ethnic group, so that the agent is considered as being punished by other members of the same group. We have demonstrated that in the framework of this principle interethnically cooperative strategies are already emerging. This fact is simply interpreted by evolutionary stability theory, those agents that have noncooperative interethnic parts of strategies extinct as a consequence of the fact that the punishment is not distinguished by its origin from intra- or interethnic interactions. For an emergence of interethnic cooperation in a population endowed by the principle of personal responsibility plays important role a fact that punishments, which originated from interethnic defections, is transferred to intraethnic interactions, where strategies that were defecting within interethnic interactions embodies smaller fitness and therefore are forced to extinction.

A major goal of our simulation studies of cooperation between two ethnic groups is a better understanding of approaches intended to promote the cooperation and of factors that may affect its breakdown. We studied two different mechanisms that can be responsible for the emergence of cooperation between two ethnic groups. Both candidates for a mechanism of cooperation were carefully studied from the point of view of evolutionary stability, whether they persist in the course of population evolution or not. Moreover, these evolutionary stability studies offer usually a "microscopic" insight into mechanisms of interactions of different strategies and help to explain why a given emerged strategy is evolutionary stable. In our forthcoming studies we would like to consider ethnic groups that are spatially structured and different mechanisms of migration of intruder agents into an ethnic group.

As the motto of the paper by Fearin and Laitin [6] a situation was described from David Laitin's childhood, when the Jewish kids were attacked by unidentified Italian kids from a neighboring school. After an unofficial complaint the situation was solved by the other school principal, who found out and punished the culprits. This situation inspired us to our model of "personal responsibility", which proved to work well in our computer simulations.

Perhaps, the results of this work might serve as a support for the general rule that an interethnic cooperation spontaneously emerges only if all the individuals from both ethnics are personally responsible for an interethnic cooperation. In this case the defective individual would be punished by its own ethnic. After our results we know, that this evolutionary stable mode of interethnic interactions is able to ensure the emergence of interethnic cooperation.

Acknowledgments. This work was supported by the grants # 1/7336/20 and # 1/5229/98 of the Scientific Grant Agency of Slovak Republic.

References

1. Axelrod, R.: *The Evolution of Cooperation*. Basic Books, New York, NY (1984)
2. Axelrod, R.: *The Complexity of Cooperation. Agent-Based Models of Competition and Collaboration*. Princeton University Press, Princeton, NJ (1997)
3. Bendor, J., Swistak, P.: The Evolutionary Stability of Cooperation. *American Political Science Review* **91** (1995) 290-307
4. Bendor, J., Swistak, P.. Types of Evolutionary Stability and the Problem of Cooperation. *Proc. Natl. Acad. Sci. USA* **91** (1995) 3596-3600
5. Bendor, J., Swistak, P.: Evolutionary Equilibria: Characterization Theorems and Their Implications. *Theory and Decision* **45** (1998) 99-159

6. Fearon J. D., Laitin, D. D.: Explaining Interethnic Cooperation. *American Political Science Review* **90** (1996) 715-735
7. Fogel, D. B.: *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The IEEE Press, New York, NY (1995)
8. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Reading, MA (1989)
9. Hofbauer, J., Sigmund, K.: *Evolutionary Games and Population Dynamics*. Cambridge University Press, Cambridge, UK (1998)
10. Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI (1975)
11. Kvasnicka, V., Pospichal, J.: Evolutionary study of interethnic cooperation. *Advances in Complex Systems* **2** (1999) 395-421
12. Lindgren, K.: Evolutionary Phenomena in Simple Dynamics. In Langton, C.G., Taylor C., Farmer J.D., Rasmussen, S. (eds.): *Artificial Life II* . Addison-Wesley Publishing Company, Redwood City, CA (1992)
13. Maynard Smith, J.: *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, UK (1982)
14. Nowak, M.A., Sigmund, K.: Tit for Tat in Heterogeneous populations. *Nature* **355** (1992) 250-253
15. Nowak, M.A., Sigmund, K.: Win-Stay, Lose-Shift Outperforms Tit For Tat. *Nature* **364** (1993) 56-58
16. Nowak, M.A., Sigmund, K.: Evolution of Indirect Reciprocity by Image Scoring. *Nature* **393** (1998) 573-577
17. Weibull J. W.: *Evolutionary Game Theory*. MIT Press, Cambridge, MA (1997)
18. Trivers, R.: The Evolution of Reciprocal Altruism. *The Quarterly Review of Biology* **46** (1971) 35-57

Author Index

Afsarmanesh, H. 335
Bochmann, O. 317
Camarinha-Matos, L.M. 335
Charlton, P. 98
Doran, J. 383
Driessens, K. 271
Durfee, E.H. 118
Fischer, K. 365
Funk, P. 365
Gasser, L. 1
van der Hoek, W. 50
Kazakov, D. 246
Kelemen, J. 281
Kollingbaum, M. 317
Kraus, S. 150
Kudenko, D. 246
Kvasnička, V. 415
Labrou, Y. 74
Mařík, V. 211
Nebel, B. 404
Oliveira, E. 173
Pěchouček, M. 211
Petta, P. 301
Poslad, S. 98
Pospíchal, J. 415
Pynadath, D.V. 187
Ruß, C. 365
Štěpánková, O. 211
Sycara, K. 17
Tambe, M. 187
Trappl, R. 301
Valckenaers, P. 317
Van Brussel, H. 317